SERIE : B
N° D'ORDRE : 60
N° DE SERIE : 51

THESE

présentée devant

L'UNIVERSITE DE RENNES

U. E. R. Structure et Propriétés de la Matière

pour obtenir

Le titre de Docteur - Ingénieur

par

G. LELARGE

Aspect fonctionnel de l'Unité de Marquage et de Signalisation de la maquette de commutateur de données HERMES

Soutenue le 14 novembre 1975 devant la Commission d'Examen

MM.	P.Y. ARQUES	Président
	C. BRIE R. GERBER J.C. MERLIN J.M. GAUTHIER	Examinateurs

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41551505_0014

SERIE : B
N° D'ORDRE : 60
N° DE SERIE : 51

THESE

présentée devant

L'UNIVERSITE DE RENNES

U. E. R. Structure et Propriétés de la Matière

pour obtenir

Le titre de Docteur - Ingénieur

par

G. LELARGE

Aspect fonctionnel de l'Unité de Marquage et de Signalisation de la maquette de commutateur de données HERMES

Soutenue le 14 novembre 1975 devant la Commission d'Examen

MM.	P.Y. ARQUES	Président
	C. BRIE R. GERBER J.C. MERLIN J.M. GAUTHIER	Examinateurs

UNIV.
LEIDEN
BIBL.

UNIVERSITE DE RENNES

-=-=-=-=-=-=-=-

U.E.R. SCIENCES ET PHILOSOPHIE

-=-=-=-=-=-=-=-

Doyens Honoraires

M. MILON Y.
M. TRHIN R.
M. SCHMITT M.
M. LE MOAL H.
M. MARTIN Y.
M. BOCLE J.

Professeurs Honoraires

M. FREYMANN R.
M. HAGENE P.
M. ROHMER R.
M. TREHIN R.
M. SALMON-LEGAGNEUR F.
M. des ABBAYES H.
M. MILON Y.

Maîtres et Conférences Honoraires

M. GRILLET
M. LECRECQ A.

Mathématiques

Professeurs

M. GUERINDON Jean
M. METIVIER Michel
M. GIORGIUTTI Italo
M. TOUGERON Jean-Claude
M. MIGNOT Alain
M. HOUDEBINE Jean
M. KEANE Michaël
M. BOULAYE Guy

Maîtres de Conférences & Chargés de Cour

M. BERTHELOT Pierre
M. BREEN Lawrence
M. CAMUS Jacques
M. CONZE Pierre
M. FERRAND Daniel
M. GUIVARC'H Yves
M. LE GOUPIL Jean
M. LERMAN Israël
M. NEDELEC Jean-Claude
M. TRILLING Laurent
M. VERJUS Jean-Pierre

Physique

Professeurs	Maîtres de Conférences & Chargés de Cour
M. VACHER Michel	M. GUIDINI J.
M. VIGNERON Léopold	M. DECAMPS E.A.
M. LE BOT Jean	
Mem ROBIN née SALOMOND	
M. ROBIN Stéphane	
M. LE MONTAGNER Serge	
M. MEVEL Jean-Yves	
M. MEINNEL Jean	
M. REGENSTREIF Edouard	
M. BRUN Pierre	
M. LEROUX Emile	
M. DUBOST Gérard	
M. ARQUES Pierre-Yves	
M. HAEUSLER Claude	
M. NUSIMOVICI Michel	

Chimie

M. VENE Jean	M. TALLEC André
M. LE MOAL Henri	
M. LEVASE	
M. PELTIER Daniel	
M. PRIGENT J.	
M. FOUCAUD André	
M. LANG Jean	
M. CARRIE R.	
M. GUERILLOT Claude-Roger	
M. KERFANTO Michel (E.N.S.C.R.)	
M. DABARD René	
M. GRANDJEAN Daniel	
M. LUCAS Jacques	
M. MARTIN Guy	

Chimie Biologique

M. JOUAN Pierre
M. DUVAL Jacques

Géologie

M. COGNE Jean	M. HAMEURT Jean-Marie
Melle DURAND Suzanne	M. LARDEUX Hubert
M. BOILLOT Gilbert	

Zoologie

Professeurs

M. RICHARD Gaston
M. MAILLET Pierre
M. RAZET Pierre
M. FOLLIOT Roger
M. ALLEGRET Paul
M. MANIEY Jacques
M. JOLY Jean-Marie
M. LEFEUVRE J.C.

Maîtres de Conférence & Chargés de Cours

M. CALLEC Jean-Jacques

Botanique

M. VILLERET Serge
M. CLAUSTRES Georges
Melle GOAS Gabrielle
Melle GOAS Marie
Mme LEMOINE Cécile

M. TOUFFET Jean

Philosophie

M. ORTIGUES Edmond

M. JACQUES Francis
M. BEYSSADE Jean-Marie

A Monsieur le Professeur P-Y ARQUES j'adresse mes remerciements les plus sincères pour l'intérêt qu'il a bien voulu porter à cette étude et pour l'honneur qu'il me fait de bien vouloir présider ce jury.

Que Monsieur le Professeur R. GERBER et Monsieur C. BRIE veuillent bien accepter ma plus sincère reconnaissance pour les conseils qu'ils ont bien voulu nous dispenser tout au long de cette étude.

Je tiens particulièrement à exprimer ma sincère reconnaissance à Monsieur J.C. MERLIN, Ingénieur en Chef des Télécommunications, Adjoint au Chef du Secteur Téléinformatique et Réseaux Spéciaux du Centre National d'Etudes des Télécommunications, et à Monsieur J.M. GAUTHIE Ingénieur des Télécommunications chargé du Département "Transmission de Données et Réseaux Privés" de l'aide qu'ils ont bien voulu m'apporter dans mes travaux.

Enfin je remercie toutes les personnes du Département "Transmission de Données et Réseaux Privés"qui ont prêté leur concours à la réalisation de ce projet.

I - PRESENTATION DU CONTEXTE

I.1 - Résumé

L'Unité de Marquage et de Signalisation de la maquette de commutateur de données HERMES est le fruit d'un travail d'équipe. Cette étude fait l'objet de deux documents séparés, l'un axé sur les problèmes de structuration, l'autre sur les aspects fonctionnels. Ces deux documents ont un chapitre d'introduction en commun.

Le présent ouvrage comporte quatre chapitres, une conclusion et des annexes :

I - Présentation

II - Relations fonctionnelles

III - Théorie de la table de décision

IV - Programmation.

L'ensemble de la présente étude et de l'étude complémentaire a donné lieu au dépôt d'un brevet enregistré sous le numéro PV 75 - 27 633.

CHAPITRE I :

Il est commun aux deux documents et replace l'unité de marquage et de signalisation dans le cadre plus vaste du projet Hermès.

Un bref exposé des problèmes généraux rencontrés en transmission de données et des possibilités offertes à l'heure actuelle par l'Administration des PTT permet de mettre en évidence d'une part la déficience des réseaux actuels dont un seul -Caducée- offre une qualité de service acceptable, et de définir les caractéristiques nécessaires des nouveaux réseaux de données.

Le réseau Hermès est ensuite présenté dans son ensemble, c'est au réseau de données, commuté qui utilise les techniques de transmission sur supports numériques (TN1) et de commutation temporelle.

On définit ensuite plus précisément le noeud de commutation Hermès en insistant sur les relations fonctionnelles entre ses divers constituants -Réseau de connexion (RCX) et organe de commande.

L'organe de commande doit assumer des tâches de deux types :

- De type synchrone : Elles seront confiées à l'UMS qui organise autour d'un processeur microprogrammé est capable d'acquérir et de filtrer un flot de données important.
- De type asynchrone : Elles seront confiées à l'unité de traitement (UT) et consisterant en la prise en compte des informations filtrées par l'UMS. L'UT est un calculateur à programme enregistré.

-:-:-:-:-:-:-:-:-

CHAPITRE II : RELATIONS FONCTIONNELLES

Dans ce chapitre on s'efforcera de mettre en évidence les différentes voies logiques que peuvent utiliser les données échangées entre le réseau de connexion et l'organe de commande.

L'UMS est d'abord présentée comme une unité d'échange d'un calculateur standard qui recevant un flot de données de l'extérieur gère le canal d'accès à la mémoire de l'unité de traitement.

Puis on décrit la chaîne de données :

Préprocesseur - processeur microprogrammé - unité de traitement.

et on distingue plusieurs types de voies de transmission suivant la nature des informations échangées en même temps qu'on introduit la fonction "aiguillage" de l'UMS.

La procédure d'échange entre les différents blocs fonctionnels est ensuite décrite sous forme d'un algorithme qui est le mécanisme de base du programme moniteur de l'UMS.

-:-:-:-:-:-:-:-:-

CHAPITRE III : THEORIE DE LA TABLE DE DECISION

On introduit dans ce chapitre la fonction filtre de l'UMS ou plus exactement de son étage synchrone le préprocesseur.

Le préprocesseur dont le principe s'apparente à celui d'un automate à nombre fini d'états ou à une table de décision permet d'analyser le train synchrone de données à fort débit que reçoit l'organe de commande en provenance du réseau de connexion de façon à ne fournir au microprocesseur et à l'UT que les informations nécessaires à la poursuite de leurs tâches ou au lancement de nouvelles tâches.

Le principe de fonctionnement du préprocesseur est simple : toutes les 976 nanosecondes le réseau de connexion envoie à l'organe de commande un ensemble d'informations relatives à une voie ; le préprocesseur détecte les modifications des paramètres d'état de la voie et ne transmet au niveau supérieur que des données concernant des voies nécessitant réellement au traitement programmé.

Le préprocesseur fonctionne par consultation de tables et la fin du chapitre traite de la génération de ces tables, de leur optimisation et de son incidence sur la charge des autres parties de l'organe de commande.

-:-:-:-:-:-:-:-:-

CHAPITRE IV : LA PROGRAMMATION

L'UMS est essentiellement un calculateur à programme enregistré et sa programmation pose quelques problèmes de logiciel intéressants :

Le premier problème exposé dans ce chapitre est celui de la génération des programmes. En effet une microinstruction est codée sur mot de 32 éléments binaires et le codage en est suffisamment compliqué pour nécessiter le développement d'un logiciel de base.

Le premier outil de programmation développé est un microssembleur programme qui à partir d'une représentation en langage symbolique du microprogramme génère un fichier binaire translatable. Le microssembleur tourne sur l'ordinateur du centre de calcul du CNET (un Honeywell Bull 6080).

Le second outil de programmation mis au point est un éditeur de lien qui à partir de différents fichiers binaires translatables créé en fichier unique ou image mémoire prêt à être changé dans la mémoire de programme de la micromachine, l'éditeur de lien tourne également sur l'ordinateur au centre de calcul du CNET.

La dernière partie de ce chapitre détaille quelques points de la programmation de la micromachine, on y trouve en particulier l'analyse du programme moniteur de L'UMS, des modules de gestion des échanges avec l'UT, avec le préprocesseur et du système de gestion dynamique de la mémoire de travail qui sert à marquer les voies en signalisation.

-:-:-:-:-:-:-:-

I.2 - Présentation

On s'accorde en général à penser que l'ensemble des moyens de communication constitue un puissant facteur de développement tant sur le plan des rapports humains que sur le plan Scientifique et Technique.

En donnant au terme communication le sens restrictif de Télécommunication, on voit encore apparaître deux grands axes de recherche correspondant à deux besoins fondamentaux de toute société moderne.

Le premier, celui de la transmission de la parole, est connu depuis longtemps déjà et le souci de développer un réseau commuté téléphonique accessible au plus grand nombre et aussi efficace que possible, a conduit au développement de nouvelles techniques telles que la transmission sur support numérique [2] et la commutation Electronique [1] qui semblent apporter des éléments de solution aux problèmes actuel.

Le second, celui de la transmission des données ou Téléinformatique, est apparu bien après la création des réseaux commutés téléphoniques. Il fut longtemps considéré comme un problème mineur puisqu'il affectait peu les rapports humains. Cependant, la croissance extraordinaire des moyens de traitement numérique de l'information a mis en évidence ce problème qui risque de devenir aigu, car l'augmentation de la puissance de traitement des systèmes informatiques, leur dissémination, la multiplicité des applications nécessitent, à moins d'un risque de freinage économique, la mise en place de réseaux de transmissions de données adaptés aux besoins toujours croissants des utilisateurs.

C'est dans cette optique qu'est née l'idée d'Hermes, réseau commuté réservé aux transmissions de données dont la conception s'inspire fortement des nouvelles techniques en matière de transmission - entièrement numérique - et de commutation - qui sera de type temporel [5], [6].

L'étude, dont le présent ouvrage expose quelques points, se situe dans le cadre de ce projet [7], [8], [9].

Elle concerne plus précisèment la définition et la réalisation de l'Unité de Marquage et de Signalisation de la maquette de commutateur Hermes [10], [11], c'est-à-dire, d'une unité d'échange microprogrammée dont le rôle est d'assurer la liaison entre le réseau de connexion temporel (RCX) du commutateur [12], [3], [4] et son organe de commande.

La conception de l'UMS (Unité de Marquage et de Signalisation) est un travail d'équipe, on a pu mettre en évidence deux grands domaines de recherches, chacun faisant l'objet d'un document séparé.

En effet, puisque l'UMS est assimilable à un calculateur à programme enregistré, on a pu dissocier la partie "matériel", c'est-à-dire l'étude de la structuration générale et de sa réalisation, de la partie "logiciel", plutôt orientée vers les problèmes d'analyse fonctionnelle de pro-

grammation et de microprogrammation [13] , [14].

I.3 - Les réseaux de transmission de données

3.1 - La transmission des données

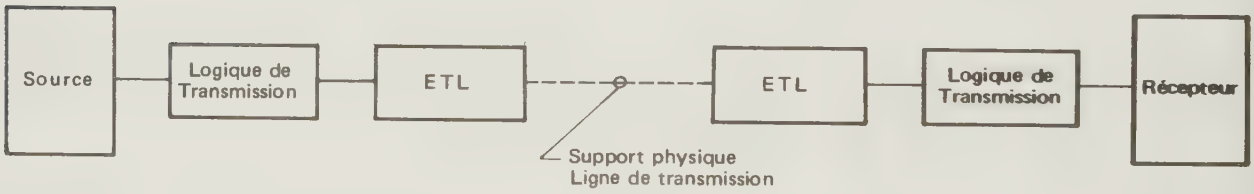
On définit la téléinformatique comme la transmission à distance de l'information reçue ou émise par un système informatique - sans en changer le contenu.

A l'apparition des techniques de traitement des données, on s'est efforcé d'implanter les centres de traitement de l'information à l'emplacement des sources d'information, mais la diversification des applications et le volume croissant des données furent bientôt tels qu'un traitement localisé devint impossible à envisager dans tous les cas.

Ainsi, s'introduisit le concept de la "liaison Téléinformatique", c'est-à-dire, de la voie de communication entre une source d'information quelconque et un dispositif récepteur distant.

3.2 - La liaison téléinformatique

Physiquement, une liaison téléinformatique est supportée par le système décrit dans le schéma ci-dessous.



Elle comprend pour un sens de transmission :

- une source d'information,
- une logique de transmission qui reçoit des informations dans la représentation interne de la source et qui produit un train binaire comprenant en général l'information émise par la source sous forme codée, avec une certaine redondance permettant la détection ou la correction des erreurs et éventuellement des informations de synchronisation telles que le récepteur puisse générer une horloge servant au décodage,
- des équipements de transmission, c'est-à-dire, un ETL (Equipement Terminal de Ligne) qui transforme le train binaire fourni par la logique de transmission en un signal adapté aux caractéristiques de la ligne de transmission.

A l'autre extrêmité de la ligne, un autre ETL reçoit le signal, régénère le train binaire et l'envoie à la logique de transmission du dispositif récepteur.

D'un point de vue logique, une liaison téléinformatique se caractérise par :

a) son débit d'information,

b) le sens de la transmission :
elle peut être simplex , c'est-à-dire unidirectionnelle, semi duplex, c'est-à-dire à l'alternat ou duplex intégral lorsque les deux sens de transmission sont utilisés simultanément,

c) le taux d'erreur, τ , défini par :

$$\tau = \frac{\text{Nombre d'éléments binaires erronés}}{\text{Nombre total d'éléments binaires transmis}}$$

3.3 - Les réseaux de transmission de données

La liaison téléinformatique est l'élément de base permettant l'élaboration de réseaux de transmission de données, c'est-à-dire d'ensembles comprenant N sources S_i.

$$\{S_i \mid i := 0,N-1\}$$

et M récepteurs R_j

$$\{R_j \mid j := 0,M-1\}$$

La solution la plus simple consiste à établir des liaisons à caractère permanent (S_i, R_j) entre les sources et les récepteurs susceptibles de communiquer entre eux; de telles liaisons sont dites liaisons spécialisées.

L'autre solution est de permettre à une source S_i d'entrer en communication non plus avec un seul récepteur R_j unique, mais avec l'un quelconque des récepteurs pris dans un sous-ensemble - groupe d'abonnés - de l'ensemble des récepteurs. Les liaisons ainsi établies ont un caractère temporaire, le réseau est dit commuté.

La voie logique (S_i, R_j) est établie à la demande de S_i ; dans ce cas, le temps d'établissement de la voie logique - ou circuit - (S_i, R_j) s'ajoute au délai de transmission dû au temps de propagation de l'information dans le réseau, et un protocole d'établissement de communication devra être mis en jeu entre, d'une part, S_i et la logique de gestion du réseau, et, d'autre part, entre cette même logique et R_j qui s'ajoute au protocole d'échange de bout en bout entre S_i et R_j après la mise en présence.

Les critères qui président à la conception d'un réseau de transmission de données, sont donc :

- le type de données à échanger,
- le délai de transmission,
- le taux d'erreur,
- la permanence de la liaison.

3.4 - Les réalisations sur le plan national

Pour faire face à la demande des utilisateurs, l'administration des PTT offre une gamme d'éléments de solution couvrant à peu près tous les besoins. Les diverses réalisations à la disposition de l'usager, peuvent être classées en trois grandes catégories :

a) les réseaux commutés non spécialisés en transmission de données,

b) les liaisons spécialisées,

c) les réseaux spécialisés en transmission de données.

a) Les réseaux commutés non spécialisés en transmission de données :

Il s'agit en premier lieu du réseau télégraphique ouvert aux transmissions de données en 1963. Il autorise deux classes de vitesses :

- jusqu'à 50 Bauds,
- jusqu'à 200 Bauds.

Durant la phase de communication, il est entièrement transparent aux codes utilisés par les

deux usagers en présence, mais la phase d'appel qui nécessite un dialogue entre appelant, appelé et réseau se fait dans le code CCITT n° 2.

Le réseau télégraphique est particulièrement utilisé pour les applications de temps partagé et en général pour toute liaison faisant intervenir des terminaux arythmiques basse vitesse.

La deuxième possibilité offerte à l'usager est celle du réseau commuté téléphonique qui offre une gamme de services allant des transmissions à faible vitesse (50 Bauds et 200 Bauds duplex) aux transmissions à moyenne vitesse (600-1200 Bauds).

L'administration loue des modems (équipements terminaux de ligne) normalisés pour les transmissions à 600 et 1200 Bauds, utilisés pour la majorité des applications de téléinformatique sur le réseau téléphonique commuté, c'est-à-dire essentiellement le temps partagé.

b) Les liaisons spécialisées :

Un usager particulier a la possibilité de louer pour son propre compte des liaisons faisant partie de l'infrastructure générale du réseau géré par l'administration. Plusieurs types de liaisons sont disponibles suivant le débit et la qualité requis pour la transmission, mais, pour une liaison donnée, deux configurations sont possibles, la ligne point à point qui ne dessert que deux abonnés et la ligne multipoint qui est partagée entre n abonnés.

Les différents types de liaisons spécialisées offertes par l'administration couvrent toute la gamme des vitesses de transmission divisées en quatre grandes catégories :

. Les liaisons télégraphiques,

. Les liaisons téléphoniques de qualité normale (600-1200 Bauds) ou de qualité supérieure permettant des transmissions à 2400-4800 et 9600 bits par seconde,

. Les liaisons en bande de base qui permettent des débits jusqu'à 72 kilobits par seconde,

. Les circuits numériques à grande vitesse qui peuvent faire appel, soit à des techniques de modulation analogique (transmission à 64 kbs/s sur groupe primaire) ou entièrement numérique, telle l'utilisation de trames numériques à 2.048 Mbits par seconde.

c) Les réseaux commutés spécialisés :

Jusqu'à présent, aucune des solutions exposées, n'offre, à la fois la gamme étendue des utilisations possibles sur liaisons spécialisées et la souplesse d'un réseau commuté, c'est ce que l'on a cherché à faire en développant des réseaux commutés spécialisés dans les transmissions de données.

La seule réalisation actuellement opérationnelle est le réseau Caducée. C'est un réseau à commutation de circuits, c'est-à-dire, qu'il y a établissement entre deux usagers en communication d'une liaison permanente pendant toute la durée de la communication. En outre, la communication est de type spatial, puisque les circuits sont créés en maintenant une continuité métallique.

En outre, Caducée offre une qualité de transmission et des services adaptés à la téléinformatique, à savoir :

- un taux d'erreur de l'ordre de 10^{-5} (il est de 10^{-4} sur le réseau téléphonique commuté),
- le temps d'établissement d'un circuit est inférieur à 500 ms,
- la sélection peut être automatique alors que cette possibilité est interdite sur le réseau téléphonique commuté.

La mise en place de Caducée correspond à la volonté d'apporter une solution rapide mais provisoire à un problème aigu.

Provisoire, car utilisant les matériels les plus sophistiqués (supports de transmission et autres de commutation).

Caducée apparaît comme un réseau de "luxe" au sens où il a fallu pousser une technique déjà éprouvée à son stade ultime de développement pour obtenir un résultat juste suffisant sur le plan des performances.

L'expérience Caducée aura au moins le mérite d'ouvrir la voie aux nouveaux réseaux de données qui devront s'appuyer sur des techniques entièrement nouvelles.

I.4 - Introduction à Hermes

On peut considérer comme acquis le fait que les nouveaux réseaux de données aurait tous au moins deux composantes communes : l'utilisation du multiplex temporel pour la transmission et de réseaux de connexion temporels

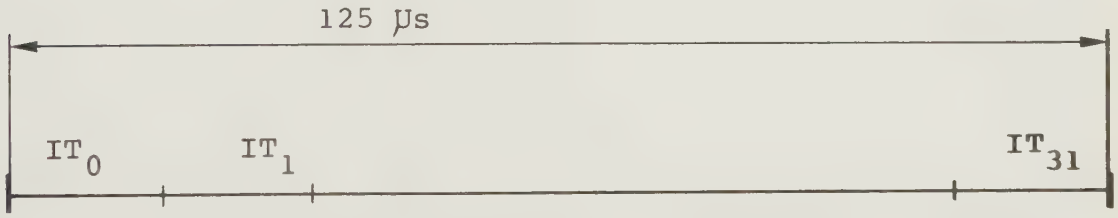
pour la commutation. Hermes n'échappe pas à cette règle, c'est pourquoi il paraît utile de rappeler ici brièvement les principes multiplexage dans le temps et la définition schématique d'un commutateur temporel.

4.1 - Le multiplexage temporel

L'idée de base du multiplexage temporel est d'associer à une voie logique de transmission (circuit) non pas une ressource physique (voie physique) continue dans le temps, mais un intervalle de temps dans une trame numérique. Ceci est utilisé tant en téléphonie numérique où l'on ne transmet plus un signal continu (information échangée entre deux abonnés en communication), mais, un signal constitué d'une suite d'échantillons quantifiés et codés sous forme de chiffres binaires, qu'en transmission de données où le signal transmis correspond aux éléments binaires échangés entre deux abonnés.

La trame numérique utilisée pour la téléphonie numérique et les transmissions de données est le multiplex à 2.048 Mb/s (appelé TN1) divisé en 32 intervalles de temps ($IT_{0,1, \ldots 31}$) ; ce multiplex fait l'objet d'une norme internationale (Avis G731, G732 du CCITT) [15].

trame numérique TN1



Pour les transmissions de données, les débits normalisés sont :

200 b/s
600 b/s
2400 b/s
9600 b/s
4800 kb/s

Or le débit sur une voie du TN1 (soit un IT) est de 2.048/32 = 64 kb/s ; on est donc contraint d'utiliser des sous-multiplex de la voie à 64 kb/s, c'est ainsi que l'on pourra considérer la voie à 64 kb/s comme un assemblage hétérogène de voies à un débit inférieur, soit par exemple :

5 voies à 12800 b/s

ou : 4 voies à 12800 b/s et 4 voies à 3200 b/s

etc ...

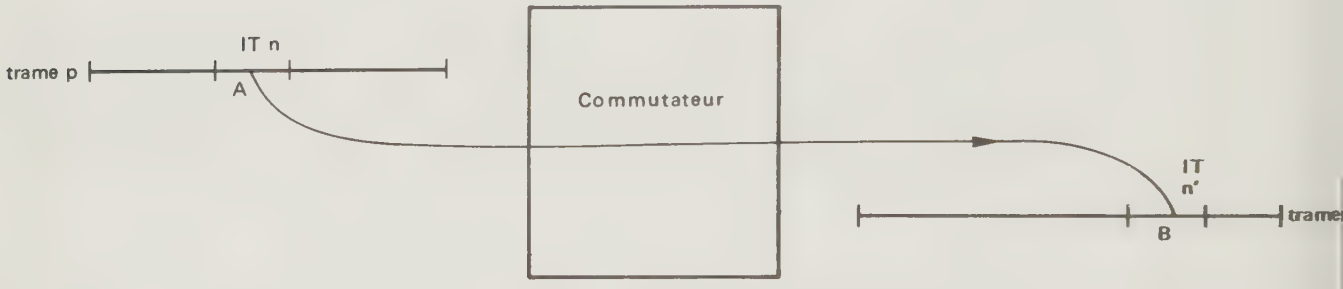
Etant entendu que la voie à 64 kb/s supporte un débit

utile de 48 kb/s,
que la voie à 12800 b/s supporte un débit utile de 9600 b/s,
que la voie à 3200 b/s supporte un débit utile de 2400 b/s.

Toutes les combinaisons sont possibles, le multiplex à 64 kb/s est défini par l'avis X50 du CCITT [15].

4.2 - La commutation temporelle

Lorsque l'on se place dans un contexte de transmission sur trames numériques, le problème de la commutation (établissement d'une voie de transmission entre deux abonnés) est celui du transfert de l'information circulant sur une voie d'une trame dans une autre voie d'une trame qui peut être différente.



L'information émise par l'abonné A, qui arrive au commutateur durant l'IT n de la trame p, doit être envoyée à l'abonné B durant l'IT n' de la trame p'. Les trames p et p', les intervalles de temps n et n' étant indépendants, il va donc falloir mémoriser les informations qui arrivent durant certains IT pour les retransmettre lors d'autres IT.

L'organe qui assure ce travail, le réseau de connexion (RCX) comprend donc principalement :

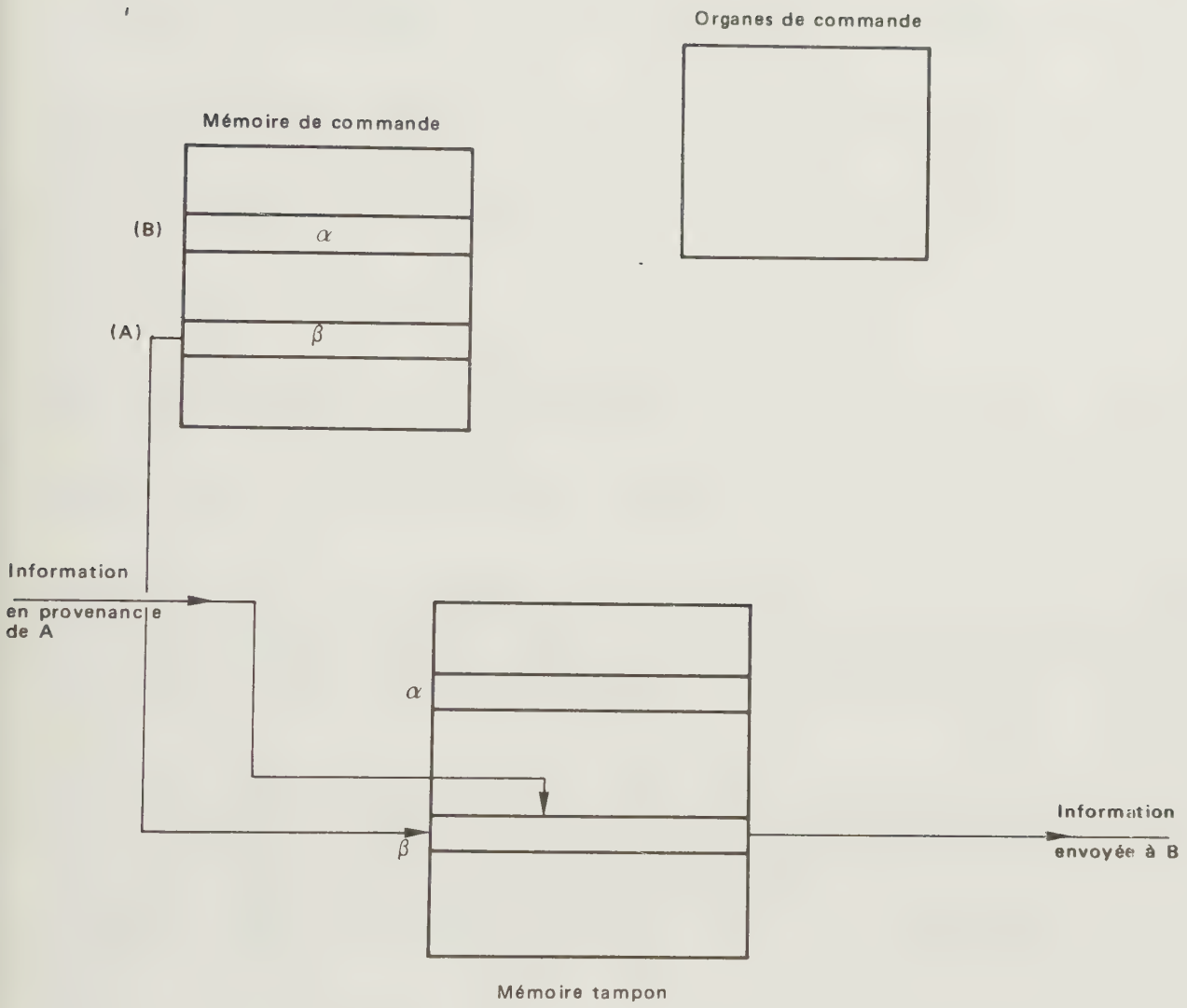
- une mémoire tampon, où l'on écrit les informations émises par les abonnés,

- une mémoire de commande, qui indique :

  . soit, la manière de vider la mémoire tampon (réseau de connexion à commande aval),
  . soit, la manière de remplir la mémoire tampon (réseau de connexion à commande amont),

- un dispositif logique qui gère cet ensemble de mémoires et en particulier qui écrit dans la mémoire de commande : les organes de commande.

La mémoire de commande contient un mot réservé pour chaque voie. Lors de l'établissement de la communication, on a écrit, sous le contrôle des organes de


Organes de commande
Mémoire de commande
(B)
α
(A)
β
Information
en provenance
de A
α
β
Information
envoyée à B
Mémoire tampon

commande, dans le mot réservé à l'abonné A, l'adresse β en mémoire tampon où l'on écrit les informations relatives à l'abonné B, et de même pour l'abonné B.

Durant la communication, les opérations effectuées dans le RCX, sont les suivantes :

- inscription de l'information émise par A en mémoire tampon à l'adresse β réservée à l'abonné B, durant l'IT n de la trame p,

- lecture lors de l'IT n' de la trame p' de la case de mémoire tampon d'adresse (β) (relative à l'abonné B),

- envoi de cette information lors de l'IT n' de la trame p' en direction de l'abonné B.

On envoie donc bien à l'abonné B, les informations émises par l'abonné A.

Lors de la rupture de la communication, les organes de commande effaceront dans la mémoire de commande, à l'adresse relative à A, le contenu de β.

Les échanges dans le sens B vers A se passent évidemment de la même manière.

La structure décrite ici correspond en fait à mettre en forme les trames sortantes dès l'écriture en MT. La lecture en mémoire tampon peut alors se faire de façon cyclique. La maquette de commutateur étudiée ici est de ce type.

Une autre solution consiste à écrire les informations en provenance de A dans un mot de la mémoire tampon à une adresse relative à l'abonné A et de venir lire ce mot lors de l'IT n' de la trame p' relatif à l'abonné B. Les trames sortantes ne sont alors mises en forme que lors de la lecture en MT. Le réseau de connexion du commutateur E 10 (PLATON) fonctionne sur ce principe.

4.3 - Structure des informations transmises

Nous avons vu que la structure du multiplex utilisé nous permettait de transmettre en ligne des octets. En fait, en plus des données échangées par les abonnés, le réseau doit acheminer deux types d'information :

- la synchronisation qui permet de reconnaître les trames et les IT,

- la signalisation qui permet l'établissement des communications.

Les informations transmises par l'abonné comporteront en plus des données, des indicateurs pré-

cisant si l'information correspond à des données envoyées au correspondant ou, au contraire, à des indications envoyées au central, lorsque l'abonné est en phase de signalisation.

En ligne, sont transmis des octets avec une structure d'enveloppe : ceci signifiant que les n éléments binaires utiles sont encadrés d'un e.b. de verrouillage (e.b.F) repérant le début de l'enveloppe et d'un e.b. d'état (e.b.S) indiquant la nature de l'information.

L'abonné (cas des abonnés synchrones) transmettant des caractères de 8 e.b., on est conduit à envisager la structure suivante, appelée quadrioctet, pour la transmission des informations envoyées par les abonnés :

octet A	F	P		S_A
" B	F	P	Q	S_B
" C	F	Q	R	S_C
" D	F	R		S_D

Les 32 e.b. transmis en ligne lors de quatre intervalles de temps successifs d'une voie, comprennent quatre octets ; chacun de ces octets comportant un e.b. de verrouillage, 6 e.b. d'information et un e.b. d'état.

En mode donnée ,les 24 e.b. d'information sont libres de tout format. Ils peuvent en particulier, comme en mode signalisation, transporter trois caractères de 8 e.b., notés P,Q et R. Les e.b. d'état S_B, S_C et S_D sont associés, chacun,à un caractère; l'e.b. S_A, par contre, est utilisé pour d'autres usages, notamment pour ce qui concerne les alarmes ou le bouclage d'un abonné.

Pour ce qui est des abonnés telex, transmettant des caractères de 5 e.b., ceux-ci sont cadrés à l'intérieur d'octets de type A.

La structure de quadrioctet ainsi envisagée nous permet de transmettre à travers le réseau des données relatives tant aux abonnés synchrones, qu'aux abonnés asynchrones.

Cette structure de quadrioctet fait également l'objet d'une norme internationale (avis X50 du CCITT) [15].

4.4 - Les services offerts sur Hermes

Pour tenir compte des besoins en téléinformatique, un certain nombre de services nouveaux seront

disponibles dans le réseau HERMES :

- numérotation abrégée : les abonnés peuvent disposer d'une liste de numéros abrégés concernant leurs correspondants les plus fréquents. Au lieu d'envoyer, lors de la phase de signalisation, tous les chiffres du numéro du correspondant, l'abonné n'enverra que les chiffres du numéro abrégé, la traduction entre les deux numéros étant faite dans le commutateur,

- appel direct (hot line) : dès le décrochage, l'abonné est mis systématiquement en communication avec un correspondant particulier,

- diffusion : l'abonné a la possibilité de transmettre, en simplex (c'est-à-dire, transmission dans un sens uniquement) les mêmes informations à plusieurs correspondants,

- création d'un réseau fermé temporaire avec station maîtresse et stations esclaves,

- transfert d'appel automatique : les communications en direction d'un abonné sont systématiquement envoyées à un autre abonné,

- remise de message différée : un message envoyé par un abonné en direction d'un ou plusieurs correspondants, sera transmis à ceux-ci dès qu'ils seront libres,

- mémorisation d'un appel infructueux : lors d'un appel, le correspondant n'étant pas libre, le commutateur rappellera celui-ci dès qu'il sera libre et appellera ensuite l'abonné demandeur pour établir la communication,

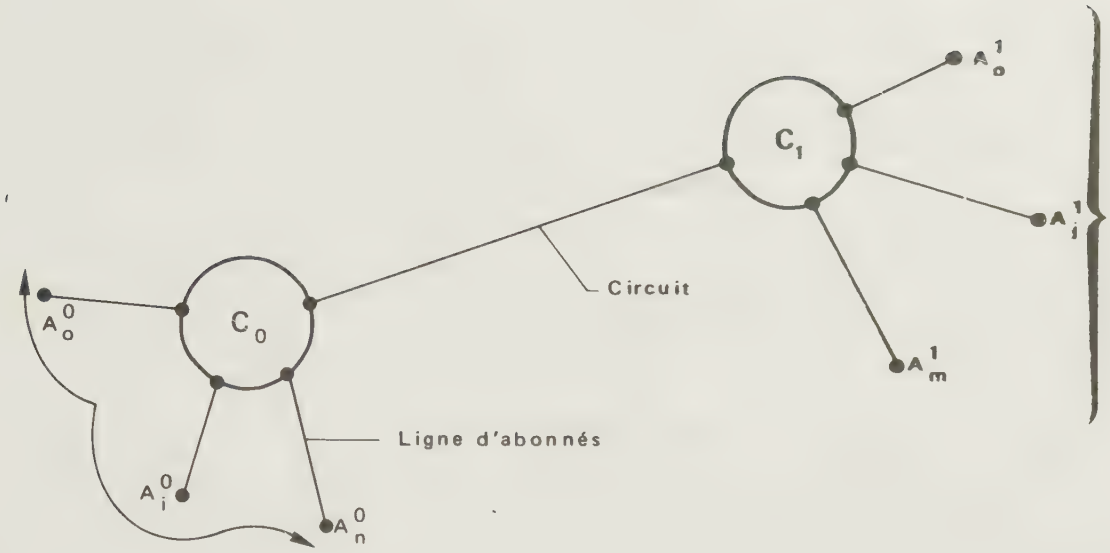
- rappel immédiat du commutateur sans libération : lors d'une communication, l'abonné a la possibilité de rappeler le commutateur sans qu'il y ait libération.

Un certain nombre de mesures seront réalisées dans le réseau en ce qui concerne le taux d'erreurs en ligne et la rapidité d'établissement des communications. Pour cette dernière, le commutateur de la maquette devra pouvoir établir cette communication en un temps inférieur à 0,5 s.

I.5 - Le Noeud Hermes

5.1 - Les accès au commutateur

D'une façon générale, un commutateur dispose de deux grandes catégories de voies d'accès ; les lignes d'abonnés directement reliées aux équipements terminaux des abonnés locaux desservis par le commutateur et les circuits qui servent aux liaisons entre commutateurs ; chacune de ces liaisons étant bidirectionnelles.



En fait, cette partition entre ligne d'abonné et circuit ne repose pas uniquement sur des considérations matérielles, mais également, sur des critères logiques, les procédures de signalisation terminale (entre un abonné et son centre de rattachement) et de signalisation inter-centraux sont différentes et le commutateur devra les distinguer.

Outre cette partition au niveau fonctionnel des moyens d'accès au commutateur, il existe une classification par type de trafic.

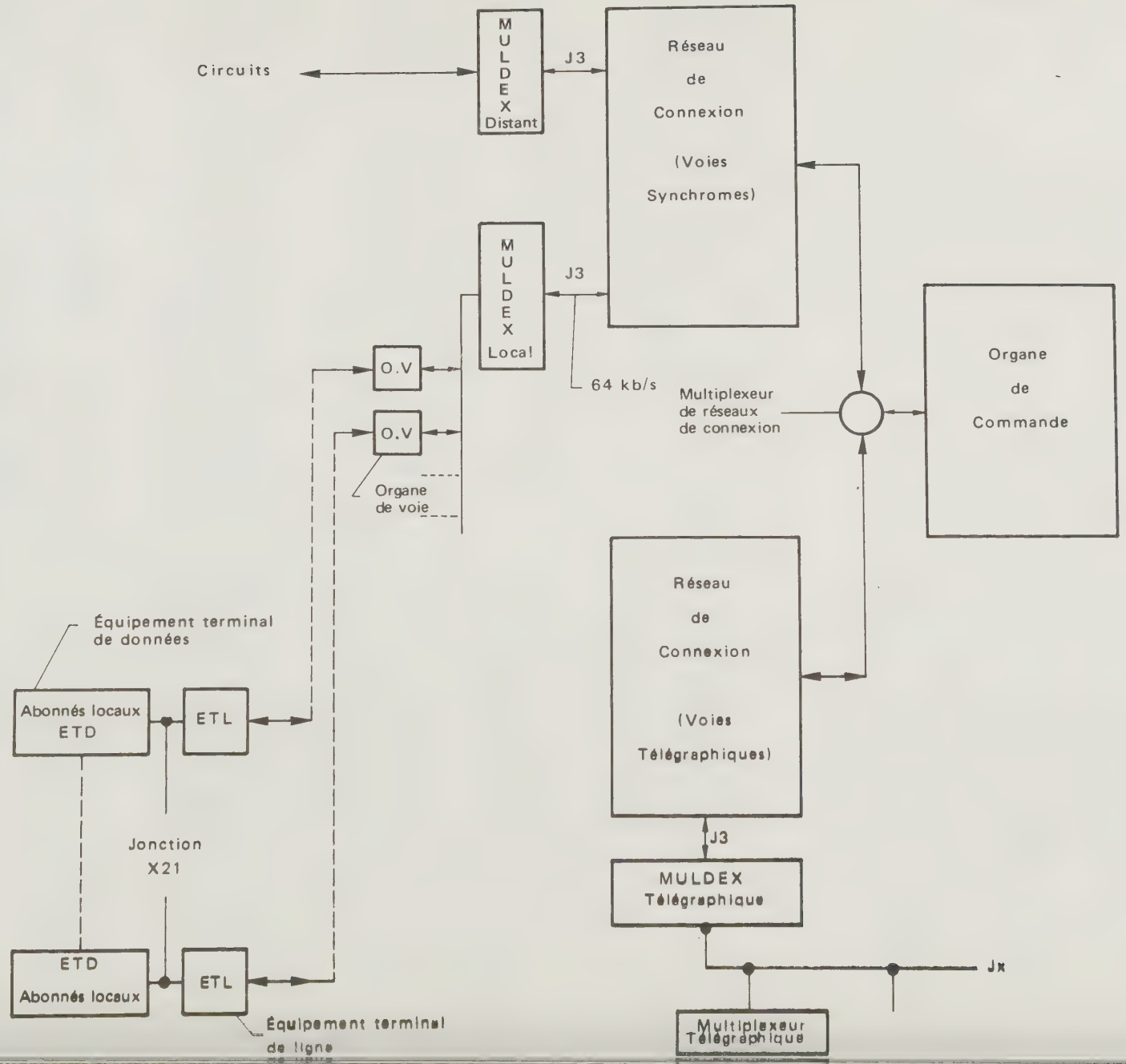
On a voulu, en effet, avec Hermes, réaliser un réseau spécialisé pour la transmission des données et offrir une gamme maximum de Services à l'usager. Les classes de trafic offertes sont pour la maquette réalisée : le 2400 bits par seconde et le 9600 bits par seconde synchrones et le 50 Bauds télégraphique.

Il y aura donc deux catégories de lignes d'abonnés : les lignes d'abonnés synchrones (2400 et 9600 bits par seconde) et les lignes d'abonnés télégraphiques; les circuits pourront être, soit spécialisés dans une classe de vitesse, dans une classe de trafic ou bien véhiculeront un multiplex hétérogène.

5.2 - Le canal d'accès au commutateur Hermes

La figure de la page suivante donne le découpage du commutateur en grands blocs fonctionnels. Pour comprendre son fonctionnement, le plus simple est de suivre le cheminement de l'information depuis l'équipement terminal situé chez l'abonné. Nous traiterons le cas des voies synchrones, le principe étant le même pour les liaisons télégraphiques.

Le nœud de commutation Hermès



- Le raccordement terminal

Vu du réseau, l'abonné est considéré comme un équipement terminal de données (ETD), celui-ci présente une jonction électrique normalisée - X26 - (du CCITT) |15| et une jonction logique - X21 -(*), qui définit la procédure de gestion de la signalisation terminale de ligne.

L'équipement terminal de ligne (ETL) est un équipement propre au réseau ; il présente également la même interface normalisée que l'équipement terminal de données (ETD), il est, par contre, entièrement transparent aux données émises par l'ETD.

Son rôle est de fournir à l'ETD un rythme (horloge bit et horloge caractère) lui permettant d'émettre son train de données. Dans le sens émission, l'ETL reçoit de l'ETD un train d'éléments binaires qu'il découpe par tranche de six, ce qui lui permet de les insérer dans un motif de quadrioctets.

Dans le sens réception, l'ETL reçoit des quadrioctets du réseau, synchronise son horloge interne d'après le motif de bits prévu et récupère dans le train de données reçu l'information à fournir à l'ETD.

Si le débit nominal de l'abonné est de 2400 bits par seconde, le débit en ligne à la sortie de l'ETL est de 3200 bits par seconde, compte tenu de l'insertion des bits F et S, de même pour un débit de 9600 bits par seconde, le débit réel en ligne sera de 12800 bits par seconde.

La transmission sur la ligne d'abonné est numérique (en code biphase).

- le raccordement de la ligne d'abonné au commutateur est fait par l'intermédiaire de l'organe de voie (ØV), celui-ci régénère le signal qu'il reçoit de l'ETL et lui envoie des quadrioctets.

L'organe de niveau immédiatement supérieur à l'ØV dans la hiérarchie d'accès au commutateur, le MULDEX (Multiplexeur - Demultiplexeur) est connecté à plusieurs ØV. Il crée à partir des différentes voies qu'il reçoit un multiplex à 64 kilobits par seconde.

Ce multiplex peut être, soit homogène, c'est-à-dire regrouper 5 voies à 9600 bits par seconde de débit nominal (5 ØV raccordés) ou bien 20 voies à 2400

(*) L'avis X21 étant en cours d'élaboration, un schéma de signalisation approximatif a été utilisé ; il est décrit en annexe.

bits par seconde (20 ØV) raccordés, soit hétérogène, c'est-à-dire, comporter des voies à différentes vitesses.

Dans le sens commutateur vers abonné, le MULDEX répartit ses différentes voies temporelles respectivement entre les différents organes de voies qui lui sont raccordés.

5.3. - Le brassage des voies

La commutation proprement dite, c'est-à-dire, puisque le commutateur est de type temporel, l'échange des données entre deux intervalles de temps des multiplex entrants se fait dans un réseau de connexion (RCX).

Un réseau de connexion reçoit plusieurs fonctions à 64 kilobits par seconde.

La maquette de commutateur actuellement réalisée dispose de deux RCX :

- le premier ou RCX "synchrone" a une capacité de 128 voies à 9600 bits par seconde, et 640 voies à 2400 bits par seconde,

- le second ou RCX "télégraphique" a une capacité de 5120 voies télégraphiques à 50 Bauds.

Le principe de fonctionnement du réseau de connexion sera exposé dans un paragraphe ultérieur, mais, on peut déjà dire que, si le RCX est capable de fonctionner de façon autonome, c'est-à-dire, de brasser les voies entrantes, il est nécessaire de lui adjoindre un organe de commande dont le rôle sera d'une part, d'établir des circuits en donnant des ordres de connexion au RCX, et, d'autre part, de superviser l'ensemble du commutateur par la gestion d'un système d'alarme.

C'est pourquoi toutes les informations provenant des multiplex entrants auxquelles les RCX ajoutent des informations caractérisant l'état de chaque voie, sont multiplexées sur un bus unique qui aboutit à l'organe de commande. Ce multiplexage se fait au niveau du multiplexeur de RCX (voir figure précédente), le bus est bidirectionnel, les échanges se font en parallèles sur 44 éléments binaires vers l'organe de commande, sur 40 éléments binaires vers le RCX ; la cadence de l'échange est fixée à 1Mhz.

· L'Organe de commande

On a vu qu'il avait à la fois un rôle actif, celui d'établir des circuits, c'est-à-dire, un lien logique entre deux voies entrantes, et un rôle de surveillance du commutateur, en particulier et du réseau, en général.

Afin d'assumer ces deux fonctions, il reçoit sur le bus de communication toutes les informations transitant dans le commutateur ainsi que des données mémorisées lui permettant de connaître l'état antérieur de chaque voie du multiplex.

L'interface entre le multiplexeur de réseaux de connexion est décrit dans [3], [4] ; pour chaque intervalle de temps, l'organe de commande reçoit :

- le caractère émis sur la voie considérée,
- l'adresse ou le numéro de la voie dans le multiplex,
- un mot permettant de caractériser l'état antérieur de cette voie,
- un groupe d'éléments binaires relatifs à la procédure d'établissement de circuits, et à la procédure de signalisation des alarmes.

La première tâche de l'organe de commande sera de filtrer cet ensemble d'informations, de façon à distinguer les voies qui nécessitent vraiment une intervention. En effet, une voie au repos (dans l'état "ligne libre") ou en communication (circuit déjà établi) est prise en compte de façon autonome par le RCX; par contre, au moment de l'établissement ou de la rupture d'un circuit, et dans le cas d'alarme détectée, l'organe de commande doit intervenir et il s'instaure un dialogue entre l'ETD trafiquant sur la voie en question et l'organe de commande.

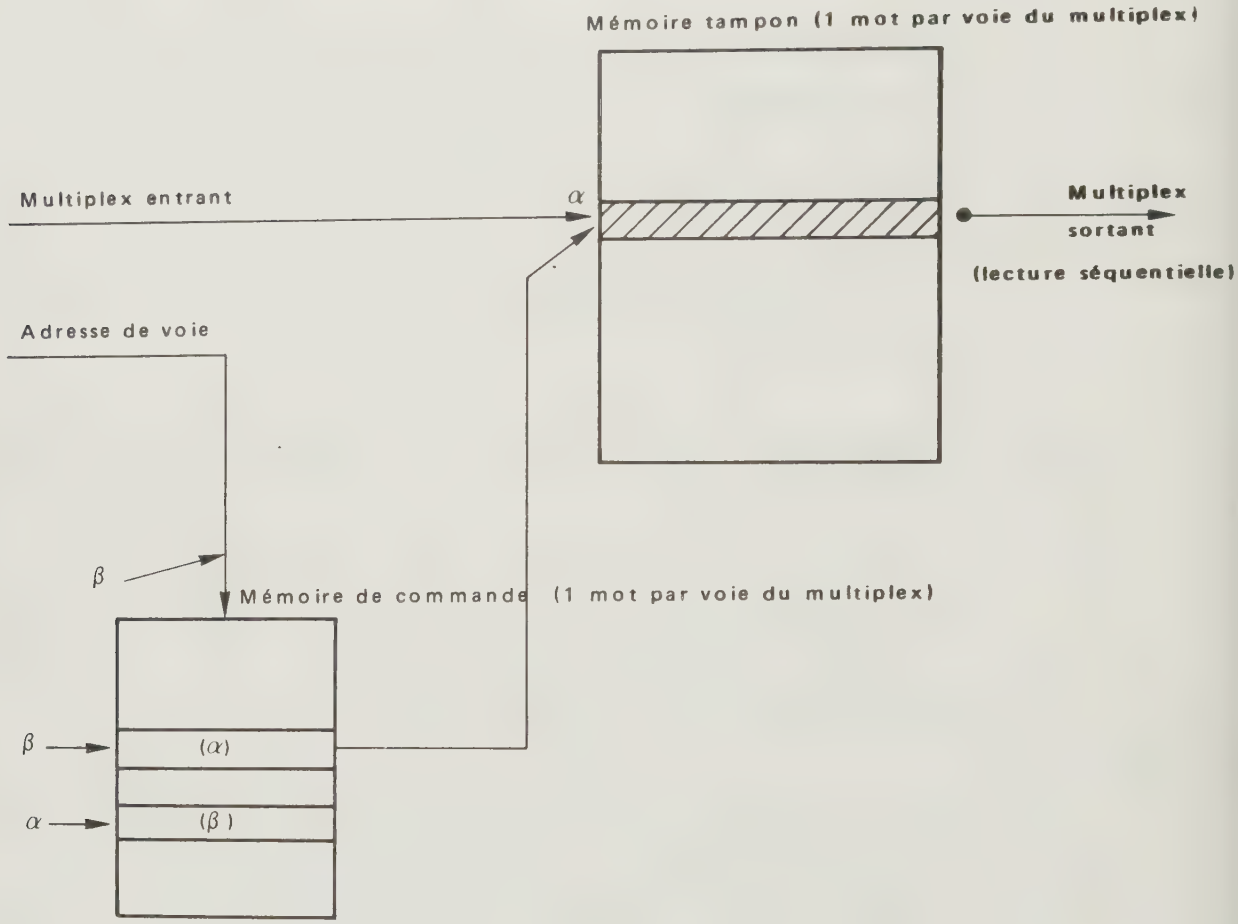
Donc, l'organe de commande a la charge de gérer la procédure d'établissement des circuits, c'est-à-dire, de recevoir les caractères de sélection de l'abonné appelant (qu'il soit local ou distant), de distinguer si l'appelé est local ou distant. Si l'abonné appelé est un abonné local, l'organe de commande devra déterminer si la communication est possible ou non ; si elle est possible, il devra entamer la procédure de mise en présence, sinon, il enverra un accusé de réception négatif au demandeur. Si le demandé est un abonné distant (cas de l'appel distant ou au transit) l'organe de commande devra trouver un circuit de départ vers son centre de rattachement, et établir un dialogue avec un autre noeud de commutation.

Il incombe, en outre, à l'organe de commande, des tâches de comptabilité, de taxation et de statistiques sur le trafic écoulé ainsi que des tâches de surveillance du réseau et du commutateur, que l'on regroupe sous le vocable de télémaintenance.

5.4 - Le réseau de connexion

- Principe et fonctionnement

Le réseau de connexion de la maquette de commutateur est dit à commande amont.



Ce type de réseau de connexion s'articule autour d'une mémoire tampon dans laquelle un mot est réservé pour chaque voie du multiplex entrant. Pour effectuer le brassage des voies, c'est-à-dire l'échange de l'information d'une voie dans une autre, ou en d'autres termes, la mise en communication de deux voies, il suffit d'écrire en mémoire tampon l'information reçue sur une voie, à un emplacement correspondant à l'adresse de la voie sortante avec laquelle la communication est établie. On crée ainsi en mémoire l'image instantanée du multiplex sortant que l'on restitue par lecture séquentielle de la mémoire tampon.

En pratique, le réseau de connexion lit dans une mémoire (mémoire de commande) - qu'il adresse par le numéro de la voie qu'il a à traiter - l'adresse d'inscription en mémoire tampon (voir figure ci-dessus).

C'est l'organe de commande qui gère le contenu de la mémoire de commande, et on voit que pour mettre en communication deux abonnés, A auquel correspond une voie d'adresse (α) et B,auquel correspond une voie d'adresse (β), il lui suffit :

- de mettre β dans le mot d'adresse (α) et α dans le mot d'adresse (β) pour qu' automatiquement le RCX assure l'entretien de la communication.

I.6 - L'organe de commande - l'UMS

L'organisation fonctionnelle du commutateur a été donnée dans le dernier paragraphe ; elle est rappelée dans le schéma ci-contre (*).

Par LRE (Ligne Réseau Entrante), le RCX reçoit des octets (de quadrioctets) en provenance d'un multiplexeur d'entrée.

Parallèlement, il dispose d'informations lui permettant de calculer l'adresse de voie de l'octet entrant, c'est-à-dire, un numéro d'identification de l'IT traité. Il distribue ensuite cette adresse sur les fils LAV (Ligne d'Adresse de Voie) aux organes de commande.

Disposant de ces informations, le RCX effectue les opérations suivantes :

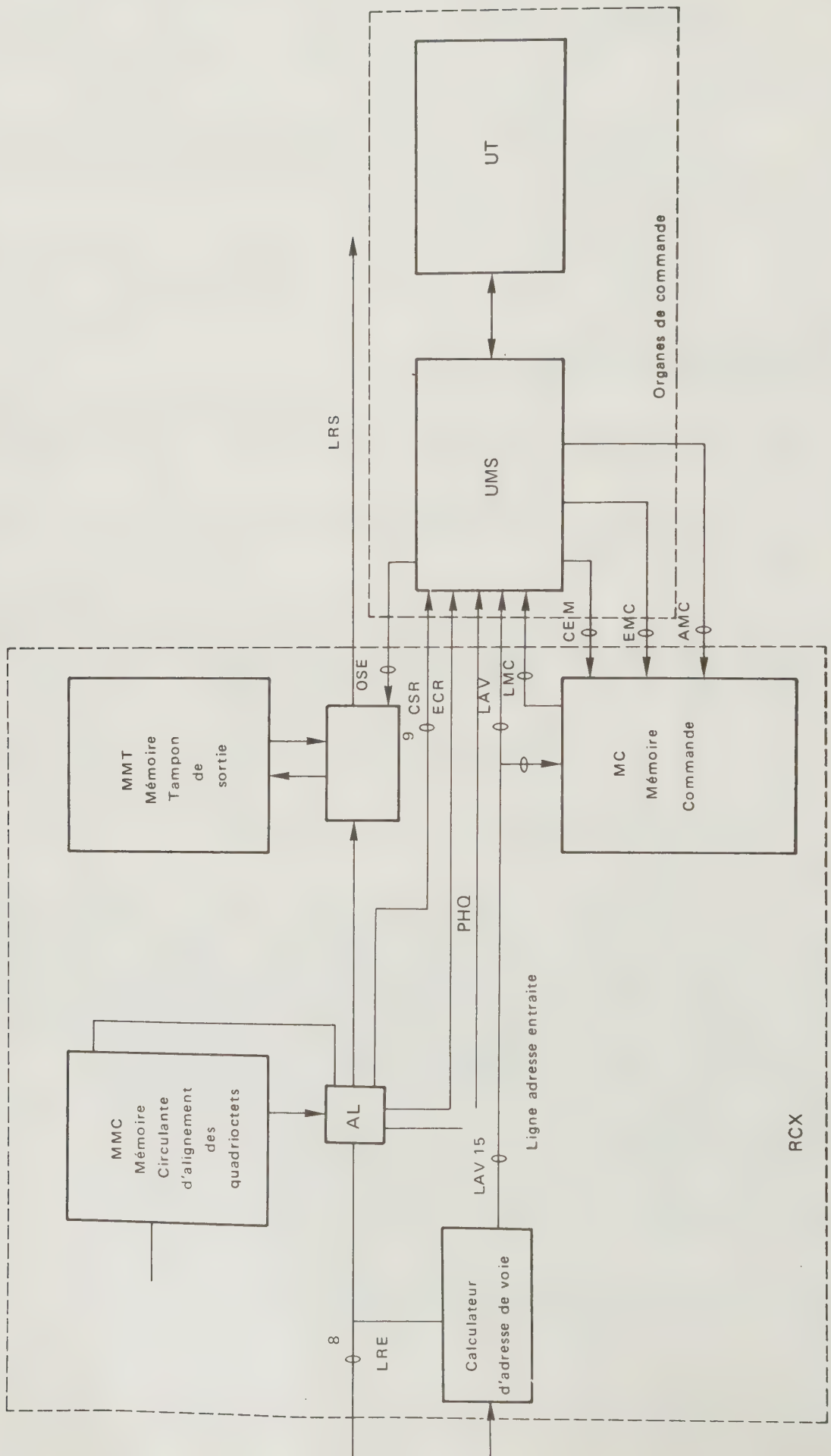
1°) Extraction des octets d'information utile ou caractères des quadrioctets

Il est facile de voir que trois fois sur quatre, on peut de deux octets successifs reçus, extraire un caractère d'information. Cette opération est réalisée au moyen d'une mémoire circulante de N mots (N étant le nombre de voies traitées par le RCX) dans laquelle on vient mémoriser le dernier octet reçu sur une voie de façon que N IT plus sort lorsqu'arrivera l'octet suivant émis sur la même voie ou dispose des deux octets successifs.

Le RCX envoie ensuite le caractère extrait aux organes de commande.

(*) Une description plus détaillée est donnée dans deux notes techniques projets [10], [11].

LES ORGANES DE COMMANDE – LEUR LIEU AVEC LE RÉSEAU DE CONNEXION



2°) Réarrangement des quadrioctets

Le problème de la mise en communication de deux abonnés A et B (transfert d'un IT_A à un IT_B) est quelque peu compliqué par le fait que A émet des quadrioctets, que B reçoit des quadrioctets et que la "phase" (numéro de de l'octet dans le quadrioctet) de l'octet présent à l'IT_A n'est en général pas la même que celle de l'octet que l'on doit émettre à l'IT_B.

En fait, on montre qu'il est possible étant donné deux octets successifs de phase quelconque, d'en extraire un octet ayant une phase donnée. Le "glissement" des quadrioctets est également réalisé à partir de la mémoire circulante.

Les octets reconfigurés sont ensuite inscrits dans la mémoire tampon.

6.1 - Les organes de commande

On peut décomposer la tâche des organes de commande en six fonctions spécifiques :

① Supervision des voies entrantes - en particulier, détection des demandes d'appel,

② Prise en compte des caractères de sélection émis par l'abonné,

③ Calcul de l'adresse de voie du correspondant éventuel à partir du numéro d'annuaire envoyé par l'appelant,

④ Connexion de l'appelant et de l'appelé, c'est-à-dire, communication au RCX de deux adresses en mémoire tampon,

⑤ Maintenance automatique du centre par la surveillance d'un canal d'alarme,

⑥ Taxation des communications, observation de trafic et statistiques.

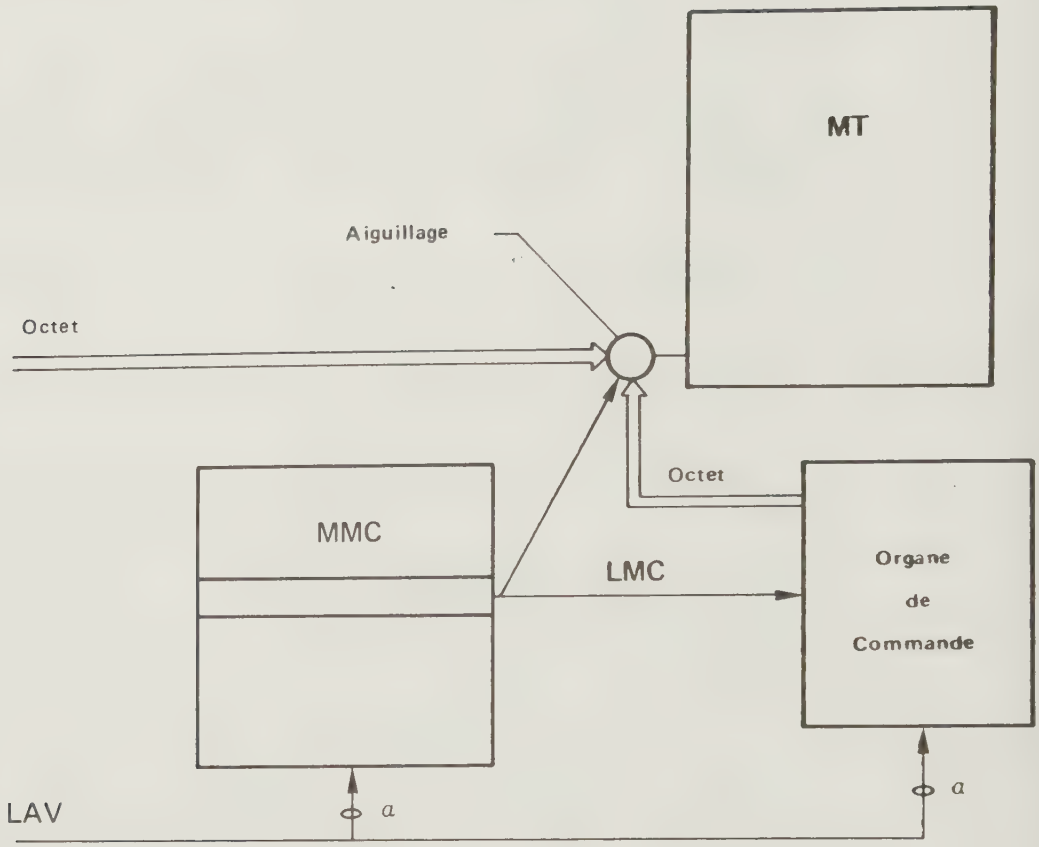
Pour mener à bien ces opérations, l'unité de commande dispose à chaque IT (toutes les 976 ns) :

- d'un caractère en provenance d'un abonné (CR) et le bit S associé ainsi qu'un indicateur de validité du caractère reçu (ECR). En effet, une fois sur quatre, le caractère émis par le RCX est sans signification,

- d'un mot de commande - LMC (Lecture Mémoire de Commande). A chaque IT, le RCX adresse une mémoire dans laquelle chaque abonné dispose d'un mot contenant des indications sur son état (libre, indisponible ...).

  Cette mémoire est utilisée par l'organe de commande puisqu'automatiquement le mot lu lui est envoyé à cha-

que IT mais il peut aussi y inscrire des informations notamment au moment de la connexion de l'organe de commande inscrit dans les mots de commande respectifs de l'appelant et de l'appelé respectivement les adresses de voie de l'appelé et de l'appelant et un indicateur de mise en communication,



- d'une base de temps, BT, donnant la cadence des caractères entrants (la fréquence choisie pour BT est à l'heure actuelle de 1.024 Mhz).

Dans l'autre sens, l'organe de commande peut :

- dialoguer avec les abonnés en allant écrire directement des octets (de quadrioctets) en mémoire tampon ; pour cela il dispose d'une indication de phase (PHQ) fournie par le RCX et d'un accès à la mémoire tampon ØSE (Octet de Signalisation Emis).

  Toutefois, l'octet émis par l'organe de commande ne sera écrit en mémoire tampon que si l'abonné correspondant n'est pas déjà en communication.

- modifier le contenu d'un mot de la mémoire de commande, en activant le fil de commande écriture mémoire (CEM)et en affichant l'adresse du mot à modifier sur les fils AMC (Adresse Mémoire de Commande).

On s'impose également de n'émettre au plus qu'un seul octet sur ØSE et de n'effectuer au plus qu'une seule écriture en mémoire de commande au cours d'un IT.

En définitive, si l'on fait le bilan des tâches incombant aux organes de commande, on peut dégager deux grandes classes de fonction :

1°) Des tâches synchrones :

A chaque IT (soit, toutes les 976 nanosecondes) l'organe de commande reçoit du RCX un ensemble d'informations relatives à un abonné ; il lui faut alors, pendant ce laps de temps :

. déterminer l'état de cet abonné (par la lecture de son mot de commande),

. lire le caractère qu'il envoie et éventuellement, déceler un changement d'état,

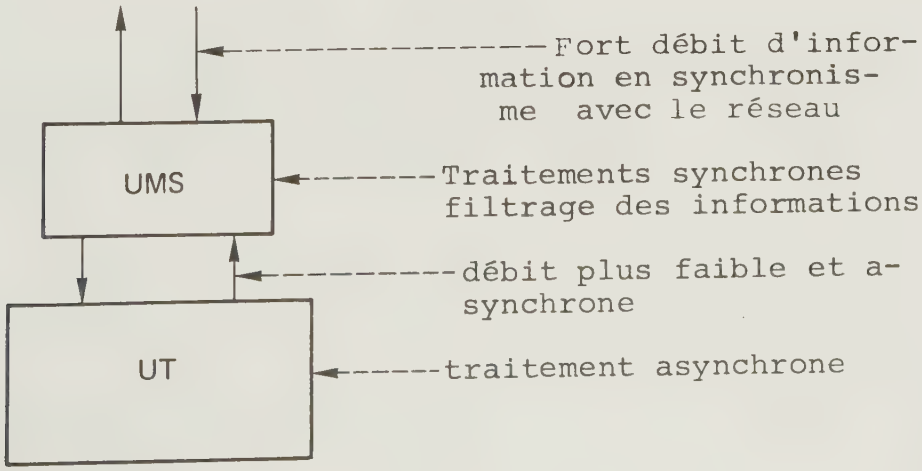
. émettre vers cet abonné un octet avec une phase donnée,

. déterminer la nature du traitement à effectuer si besoin est (changement d'état - initialisation d'une communication

2°) Des tâches asynchrones :

Outre les tâches synchrones qui doivent être effectuées à une cadence imposée par le réseau, l'organe de commande se voit confier des travaux moins urgents, et plus sophistiqués tels que la gestion d'un fichier d'abonnés, le calcul de l'adresse de la voie d'un abonné demandé.

C'est de cette dualité au niveau de l'analyse fonctionnelle qu'est apparu le besoin de scinder l'organe de commande en deux (figure ci-dessous) :



a) L'Unité de Marquage et de Signalisation (UMS) chargée des traitements synchrones. C'est elle qui assure le dialogue entre le RCX et l'organe de commande, elle joue le rôle d'unité d'échange entre le RCX et l'unité de traitement.

Elle supervise les communications par surveillance de l'état de chaque abonné ; elle détecte les alarmes éventuelles en provenance des autres organes du commutateur.

b) L'Unité de Traitement :

Elle est chargée des traitements asynchrones ; son rôle de base est, compte tenu des caractères de sélection émis par un abonné appelant (retransmis par l'UMS) de :

. calculer l'adresse de voie du correspondant,

. vérifier si l'appelant peut accéder à l'appelé, c'est-à-dire :

- si l'appelé n'est, ni occupé, ni indisponible, ni en dérangement ... ,

- si l'appelant a accès à la classe de service demandé,

. donner à l'UMS les directives pour l'établissement de la communication, qui en assurera l'exécution.

En outre, l'UT (Unité de Traitement) devra assumer la gestion du centre, c'est-à-dire, la taxation et l'observation de trafic.

Compte tenu de la complexité des opérations, l'unité de traitement doit être un calculateur à programme enregistré (un modèle T 1620 de la Télémécanique a été choisi) de type classique. Pour l'UMS, le problème n'est pas aussi simple; d'une part, sa tâche est suffisamment complexe pour nécessiter la mise en oeuvre d'un calculateur à programme enregistré, et, d'autre part, les contraintes de temps imposées par le volume des informations à traiter écartent à priori l'hypothèse d'un calculateur de type classique. C'est pourquoi une structure originale construite autour d'une unité centrale microprogrammée - donc rapide - à laquelle on adjoint un automate programmable pour le rôle de filtre, a dû être développée.

II - RELATIONS FONCTIONNELLES

II.1 - L'UMS - Une unité d'échange

Lorsqu'il s'est agi de fixer définitivement la structure de la maquette du commutateur Hermes, il est apparu nécessaire que l'organe de commande de ce commutateur soit un calculateur à programme enregistré et comme il ne saurait être question dans un premier temps au moins de concevoir un calculateur spécialement adapté à ce genre de problème, il a donc été décidé que l'organe de commande serait un ordinateur de type universel.

Un modèle T 1600 de la Télémécanique a été choisi ; c'est un petit calculateur à mots de 16 e.b. dont le temps moyen d'instruction est voisin de 4 microsecondes et dont la mémoire vive a un temps d'accès de l'ordre de 750 nanosecondes.

Un ordinateur est un organe de traitement essentiellement arythmique, c'est-à-dire, que, d'une part, il peut être interrompu par des actions extérieures et que, d'autre part, s'il fonctionne en mode acquisition de données, on ne peut lui imposer d'être disponible pour prendre en compte une nouvelle donnée à des instants fixes.

C'est précisèment le cas de l'organe de commande puisque le RCX lui envoie à une cadence de 1.024 Mhz, soit toutes les 976 nanosecondes, un ensemble d'informations relatives à une des voies d'un multiplex entrant, et, pendant ce même laps de temps, l'organe de commande doit renvoyer au RCX au moins un octet de signalisation.

La Jonction entre le réseau de connexion et l'organe de commande est décrit dans [3] et [4].

Le problème ainsi posé était insoluble à moins de disposer entre le RCX et l'unité de traitement de l'organe de commande d'un étage qui joue un double rôle :

- le rôle d'unité d'échange,

- le rôle de filtre.

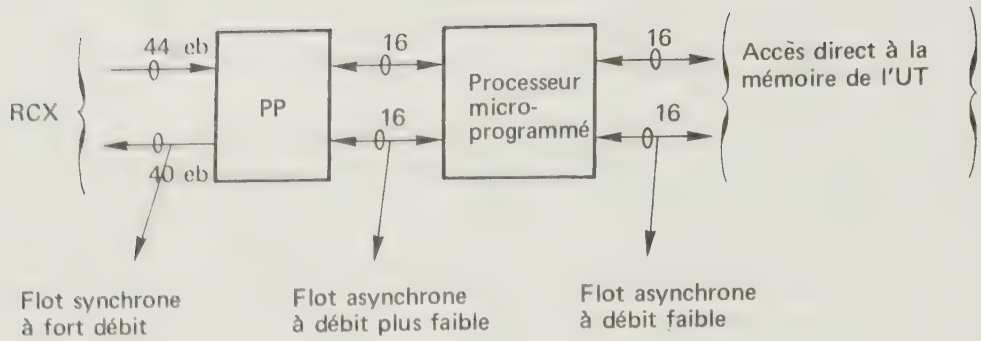
La structure de cet organe particulier (l'UMS) est abondamment décrite dans le chapitre I du document concernant la structuration où l'on montre que l'organe de commande doit avoir trois composantes :

- un automate câblé : le prépocesseur,

- une unité de traitement microprogrammée,

- une unité de traitement.

Les deux premiers étages constituent l'UMS.



Parmi les trois composantes constituant l'organe de commande, une seule - le prépocesseur, a un fonctionnement synchrone, donc au sein même de l'organe de commande, tous les échanges se font de façon asynchrone.

Au niveau de l'unité de traitement (et même au niveau de l'UMS) toutes les tâches programmées ont la même priorité. En effet, toutes les voies de communication, ou les circuits, ne sont pas hiérarchisés, or à un instant donné, l'UT traite des données relatives à une voie, et il n'y a aucune raison pour interrompre ce traitement et lancer une tâche relative à une autre voie. Ceci est également vrais pour l' UMS et d'une façon générale, la notion d'interruption de tâche a été autant que possible exclue dans le système que nous avons essayé de concevoir.

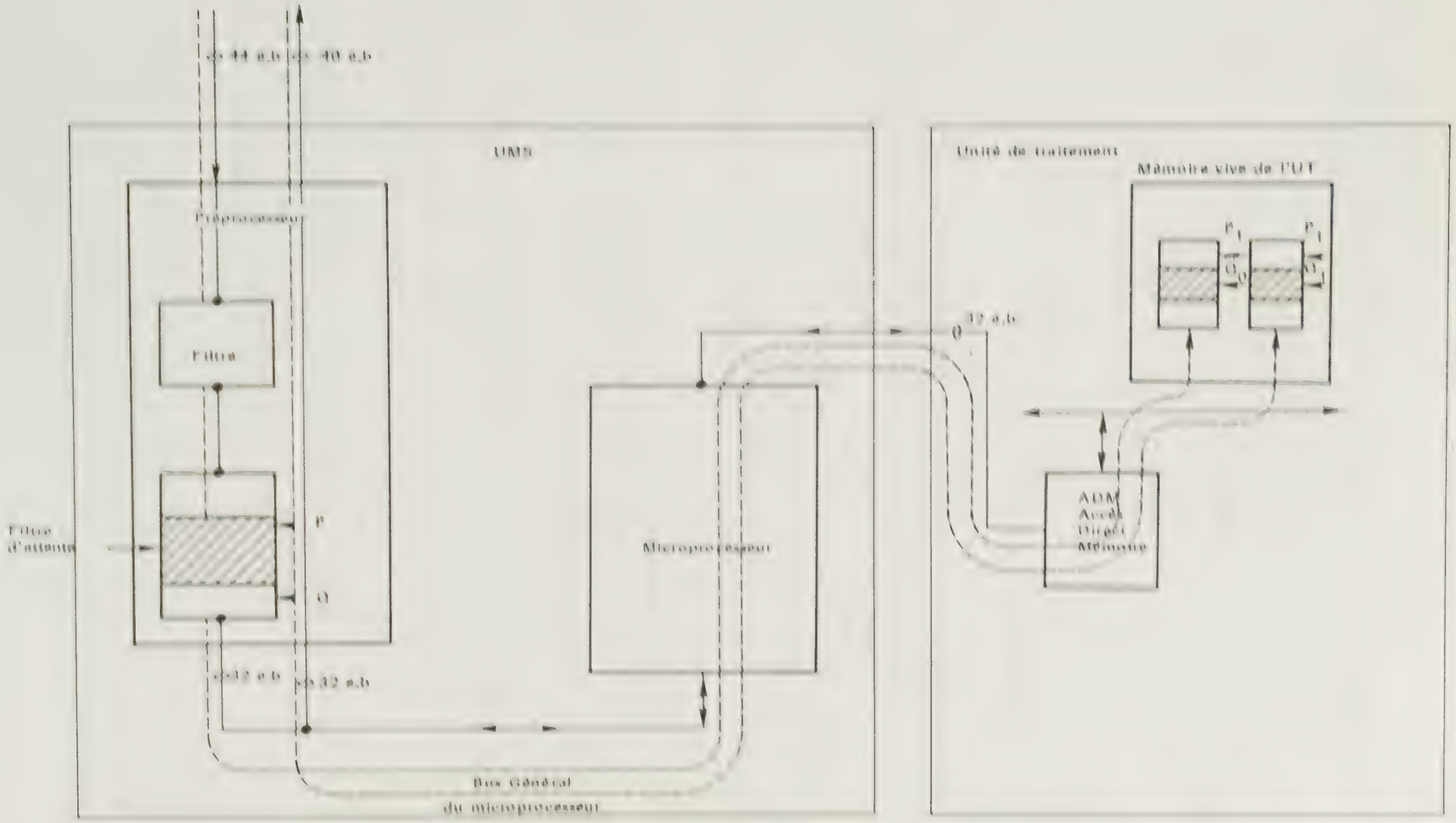
La tâche essentielle de l'UMS est de communiquer à l'UT les informations nécessaires aux traitements des voies en signalisation, or pour échanger de l'information avec un calculateur il existe deux voies distinctes :

- la première utilise l'accès normalisé au bus d'entrée-sortie et dans ce cas, l'acquisition des données se fait au point de vue du calculateur selon un mode dit mode programmé par interruption, c'est-à-dire que les points d'accès sont hiérarchisés et que lorsque l'on veut fournir une donnée au calculateur, ou obtenir une donnée au même calculateur, il faut l'interrompre pour qu'il prenne en compte la requête qu'on lui adresse,

- la deuxième voie de communication avec le calculateur est sa mémoire vive.

Dans ce cas, le périphérique (ou l'unité d'échange) inscrit directement en mémoire ou y lit (qui devient une ressource partagée) de l'information à échanger.

La notion d'interruption a été bannie autant que possible du logiciel de l'organe de commande - ceci



44 e.b
40 e.b
UMS
Préprocesseur
Filtre
Filtre d'attente
P
Q
32 e.b
32 e.b
Microprocesseur
Bus Général du microprocesseur
Unité de traitement
Mémoire vive de l'UT
32 e.b
ADM Accès Direct Mémoire

pour deux raisons, la première, déjà exposée étant qu'il était illogique d'interrompre une tâche par une tâche de même priorité, la seconde, étant qu'une interruption est coûteuse en temps machine, puisqu'il faut sauvegarder le contexte d'exécution de la tâche interrompue, afin de la réactiver ultérieurement et restaurer le contexte de la nouvelle tâche initialisée - tout ceci apporte, dans le cas du T 1600 une pénalisation de l'ordre de 50 à 70 microsecondes.

Ces deux raisons nous ont conduit à adopter pour les échanges UMS- unité de traitement, la seconde solution, celle de l'accès direct à la mémoire de l'UT.

On peut ainsi distinguer deux voies de communication qu'utilise l'information échangée entre organe de commande et réseau de connexion.

a) La première, ou voie "montante" peut s'envisager comme un système à "commande hiérarchisée" dans lequel chaque maillon de la chaîne effectue un filtrage sur l'information que lui envoie le maillon d'ordre immédiatement inférieur puis fournit à l'échelon supérieur d'une part, l'information filtrée, et, d'autre part, une commande - ou plutôt - des indications concernant la manière de traiter cette information.

La chaîne montante comprend trois maillons :
- prépocesseur - unité de traitement microprogrammée - unité de traitement :

. Le prépocesseur reçoit - à chaque intervalle de temps du multiplex de plus haut niveau - soit, toutes les 976 nanosecondes, un ensemble d'informations sous forme d'un mot de 44 e.b. relatif à l'état d'une voie d'un multiplex entrant.

. Le prépocesseur analyse systématiquement chaque contexte et décide du traitement nécessaire de cette voie.

A ce niveau, deux cas sont possibles :

- si l'action de l'organe de commande concernant la voie en cours d'analyse, ne nécessite qu'un traitement élémentaire , c'est-à-dire, l'envoi d'un octet correspondant à un caractère permanent sur la voie considérée (voir chapitre prépocesseur du document concernant la structuration) ou ne nécessite aucun traitement, alors le prépocesseur l'exécute seul de façon autonome sans en référer au niveau supérieur,

- si, par contre, l'action décidée nécessite un traitement plus complexe, alors le prépocesseur en demande l'élaboration au niveau supérieur, en fournissant au second échelon (l'unité de traitement microprogrammée) une partie de l'information reçue sur la voie à laquelle il adjoint des indications concernant la nature du traitement à effectuer de façon à former un <u>message</u> élémentaire qu'il met dans une file.

. L'une des fonctions de l'unité de traitement microprogrammée est de demander au niveau inférieur de lui fournir au coup par coup des messages.

Chaque message est décodé par programme et une action est entreprise en fonction de la commande reçue de l'étage inférieur. Cette décision peut être l'envoi à l'unité de traitement d'une commande relative à une action factuelle associée à une partie de l'information reçue (un caractère de sélection, par exemple) ou une information seule.

. L'unité de traitement - le maillon le plus élevé dans la hiérarchie - reçoit directement dans sa mémoire vive, les commandes et les informations de bout de chaîne et élabore la décision au plus haut niveau.

b) La seconde, ou voie "descendante" apparaît comme plus directe ; elle est utilisée d'une part, pour émettre des octets sur les voies en dialogue avec l'organe de commande, et, d'autre part, pour donner au RCX des ordres directement exécutables.

Dans les deux cas, la décision est prise au niveau de l'unité de traitement et transmise sous forme de commande à l'unité de traitement microprogrammée qui agit directement sur le réseau à travers le prépocesseur complètement transparent dans ce cas.

En définitive, la chaîne des données bidirectionnelles (prépocesseur - UT microprogrammée - UT) qui physiquement, apparaît comme symétrique en considérant l'interface UMS - Réseau de connexion comme une liaison "Full Duplex", devient complètement dissymétrique quand on l'envisage sous l'angle logique. En effet, la voie montante est une chaîne où la prise de décision et le filtrage de l'information sont des fonctions réparties dans les différents maillons, alors que la voie descendante est simplement une des voies de transmission d'un ordre donné par l'organe le plus élevé dans la hiérarchie et transmis de manière presque transparente au réseau.

II.2 - Algorithme de gestion des échanges

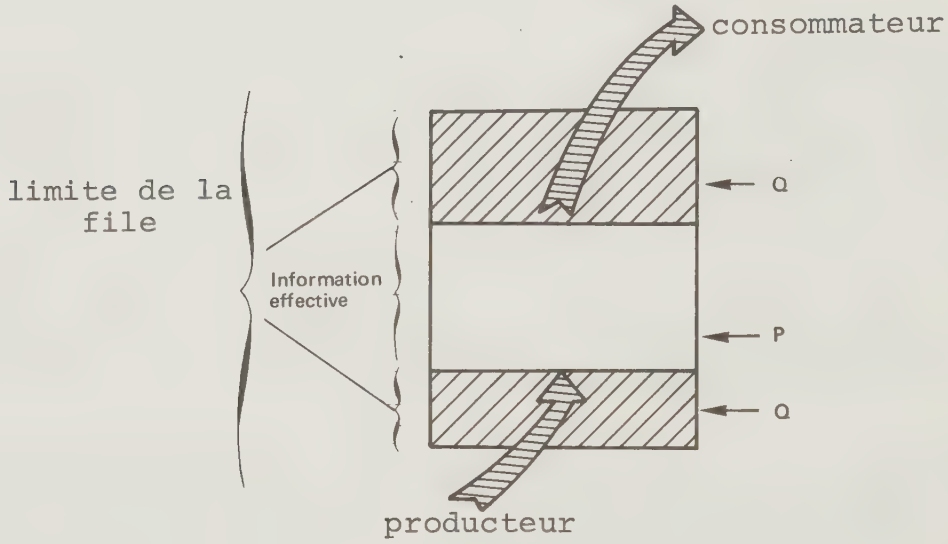
2.1 - Les processus parallèles

Si l'on se place d'un point de vue purement logiciel, c'est-à-dire, si l'on raisonne en terme de machines logiques, on peut considérer l'UMS et plus précisément la fonction unité d'échange de l'UMS, comme un ensemble de processus s'exécutant en parallèle , à savoir la fonction prépocesseur, la fonction gestion des échanges de l'UT microprogrammée, la fonction acquisition de données de l'UT.

Ainsi posé, le problème de l'échange d'information entre le RCX et l'organe de commande, se ramène

au problème beaucoup mieux connu de la communication entre processus parallèles et leur synchronisation [16] .

Dans le cas présent, les trois processus ou machines logiques sont supportés par des machines physiques distinctes ayant des caractéristiques différentes tant sur le plan de la rapidité que sur celui de la puissance ; il était donc impossible de réaliser un asservissement mutuel de ces processus. C'est pourquoi la solution retenue a été de les faire communiquer entre eux au moyen de files d'attente suivant une procédure dite : producteur-consommateur.



La figure ci-dessus, décrit le fonctionnement d'une telle file ; l'information est rangée dans la file sous forme de blocs ou mots, qui constituent le message élémentaire échangé entre le producteur et le consommateur.

La file est une ressource partagée entre producteur et consommateur ; elle est définie par deux pointeurs, P et Q. Q indique le prochain mot à écrire et P le prochain mot à lire dans la file.

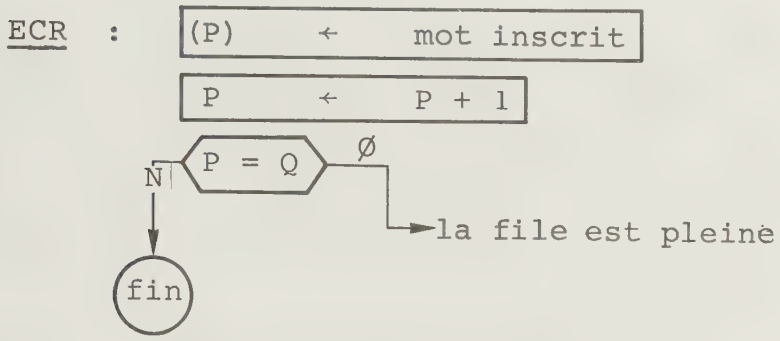
La gestion des pointeurs P et Q est réalisée au moyen de deux petits mécanismes ou primitives :

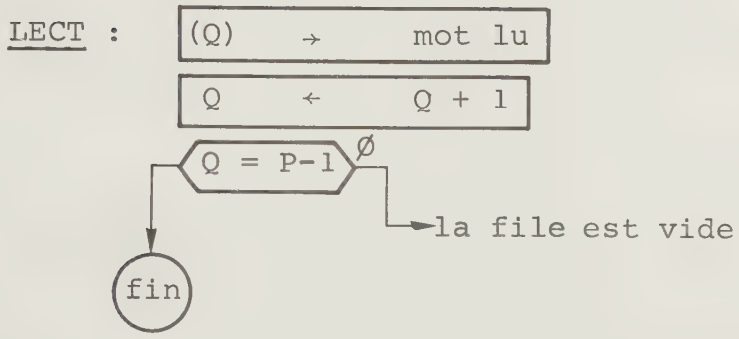
- une primitive d'écriture, ECR, qui est activée par le producteur et qui inscrit un mot dans la file,

- une primitive de lecture, LECT, qui est activée par le processus consommateur et qui lit un mot dans la file.

Ces deux primitives sont décrites page suivante ; on notera simplement que chacun des deux poin-

teurs P et Q sont définis modulo n, n étant la dimension maximum de la file d'attente.





Puisque la file d'attente ainsi définie est une ressource à double accès, et du fait que, par essence, les processus producteur et consommateur ne sont pas synchronisés, il apparaît des cas de conflits pour l'accès à cette ressource.La résolution de ces conflits peut se faire d'une façon purement logicielle, lorsque les différents processus sont supportés par la même machine physique au moyen de sémaphores d'exclusion ou de synchronisation; dans le cas précis de l'UMS, les processus étant physiquement distincts, il a fallu trouver des solutions particulières pour chaque cas. C'est pourquoi on trouve dans le système deux types de file d'attente :

- Pour les échanges entre le prépocesseur et l'unité de traitement microprogrammée, la file d'attente est une partie câblée du prépocesseur ; en particulier, les primitives ECR et LECT sont réalisées au moyen de circuits logiques. Le fait de disposer d'un mécanisme de gestion des pointeurs d'une nature (matérielle) autre que celle des processus utilisateurs (logicielle) résout de facto tout conflit d'accès.

  Ce choix particulier a été guidé par le fait que l'on ait voulu décharger l'unité de traitement microprogrammée de la gestion de cette file qui aurait consommé une partie non négligeable de sa puissance de traitement et que l'autre processus coopérant - le prépocesseur - était trop rudimentaire pour pouvoir assurer cette gestion par programme.

- Pour les échanges entre l'unité de traitement microprogrammée et l'unité de traitement (T 1600), on

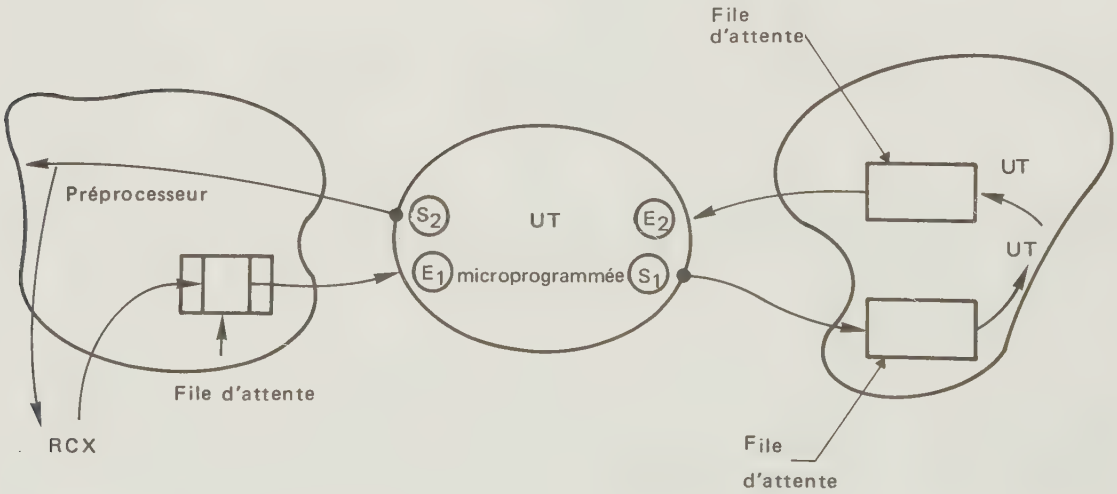
utilise deux files d'attente, l'une pour la voie "montante", l'autre pour la voie "descendante".

Donc, les deux processus coopérants sont tour à tour producteur et consommateur, selon qu'ils s'adressent à l'une ou l'autre file. Les files sont localisées dans la mémoire vive de l'unité de traitement ; toutefois, les primitives de gestion des pointeurs sont réalisées en logique câblée au moyen d'un esclave de l'unité de traitement (l'ADM qui est décrit dans le document concernant la structuration au chapitre IV) auquel s'adressent les différents processus pour accéder à la file.

Ce dispositif permet également de résoudre tous les conflits d'accès.

II.3 - La Gestion des transactions

Le paragraphe précédent a montré que le pouvoir de décision était hiérarchisé et réparti selon les trois maillons de la chaîne des données. Il faut maintenant faire la distinction entre la fonction décision qui prend en compte le résultat d'un échange d'information et la fonction échange d'information, qui est un mécanisme indépendant de la signification de l'information échangée.



Si l'on considère le schéma ci-dessus qui représente le chemin des données, on s'aperçoit que l'unité de traitement microprogrammée est le noeud par où transite toute l'information. C'est en fait elle qui joue le rôle de maître dans la gestion des échanges et déterminer l'algorithme de gestion des transactions est équivalente à définir l'organigramme du programme moniteur de ce processeur.

L'unité de traitement microprogrammée voit

deux points d'entrée E_1 et E_2 (E_1 correspond à la file d'attente d'échange avec le prépocesseur et E_2 , la file d'attente d'échange avec l'unité de traitement) et deux points de sortie S_1 et S_2. (On pourra, dans un premier temps, considérer S_1 comme l'accès à une file d'attente d'échange avec l'unité de traitement; ce point sera détaillé ultérieurement, S_2 correspond au point d'accès en sortie au prépocesseur).

On a vu au chapitre précédent que le prépocesseur en tant que machine physique était asservi par une horloge externe fournie par le RCX, donc, le processus supporté par cette machine physique est un processus synchrone, c'est-à-dire qu'il s'exécute entièrement en un cycle de l'horloge pilote soit toutes les 976 nanosecondes. D'autre part, la file d'attente d'échange avec l'unité de traitement microprogrammée a une longueur finie (256 mots) et le prépocesseur ne dispose d'aucun autre moyen de mémoriser de l'information, il impose donc une contrainte absolue au système :

R_0 : [la file d'attente d'échange entre le prépocesseur et l'unité de traitement microprogrammée ne doit jamais déborder.]

Le débordement de cette file entraîne une perte définitive d'information, ce qui est intolérable.

On verra que le prépocesseur a des moyens de se prémunir en partie contre ces débordements en mettant en oeuvre une fonction de filtrage plus sévère lorsque le remplissage de la file devient critique, mais la contrainte exposée ci-dessus demeure toujours valable.

Les autres maillons de la chaîne fonctionnent de façon asynchrone, c'est-à-dire, que chacun des deux processus peut être considéré comme une succession de tâches élémentaires dont l'enchaînement dépend du contexte d'exécution, on ne peut donc pas parler de tâches cycliques comme pour le prépocesseur ; en outre, chacun d'eux a la possibilité de stocker de l'information (l'espace disponible dans la mémoire de travail de l'unité de traitement microprogrammée est limité, mais celui de l'unité de traitement peut être considéré du point de vue de l'UMS comme infini).

D'où les deux règles suivantes :

R_1 : [l'unité de traitement microprogrammée dispose d'une capacité de mémorisation limitée]

R_2 : [l'unité de traitement dispose d'une capacité de mémorisation qui peut être considérée comme infinie.]

A la lumière de ce qui vient d'être dit, on peut déjà conclure qu'il doit exister une hiérarchie parmi les points d'échanges de l'information. En effet, il semble

que l'entrée E_1 soit prioritaire, à cause de la contrainte R_0, ensuite, par ordre décroissant, viennent S_1 à cause de R_1 - l'UT microprogrammée ne peut mémoriser longtemps l'information reçue de E_1, puis E_2 d'après R_2 - l'UT a des moyens de rétention tels que l'on peut différer la prise en compte des informations en retour de l'UT et enfin S_2.

Le fait que S_2 se trouve au niveau de priorité le plus bas peut sembler surprenant au premier abord, puisque si l'on applique la règle R_1, on doit attribuer à S_2 la même priorité que S_1 mais on s'impose en outre d'associer par couple une entrée à une sortie de façon que le même processus élémentaire soit à la fois receveur pour une entrée et serveur pour la sortie associée sans qu'aucune autre opération d'entrée-sortie n'ait lieu dans l'intervalle; un tel processus élémentaire est appelé <u>tâche</u>, d'où la règle R_3 :

R_3 : [une tâche ne peut être interrompue]

Dans notre schéma, on trouve donc deux couples d'entrée-sortie (E_1 - S_1) et (E_2 - S_2) d'où la règle des priorités expliquée précédemment.

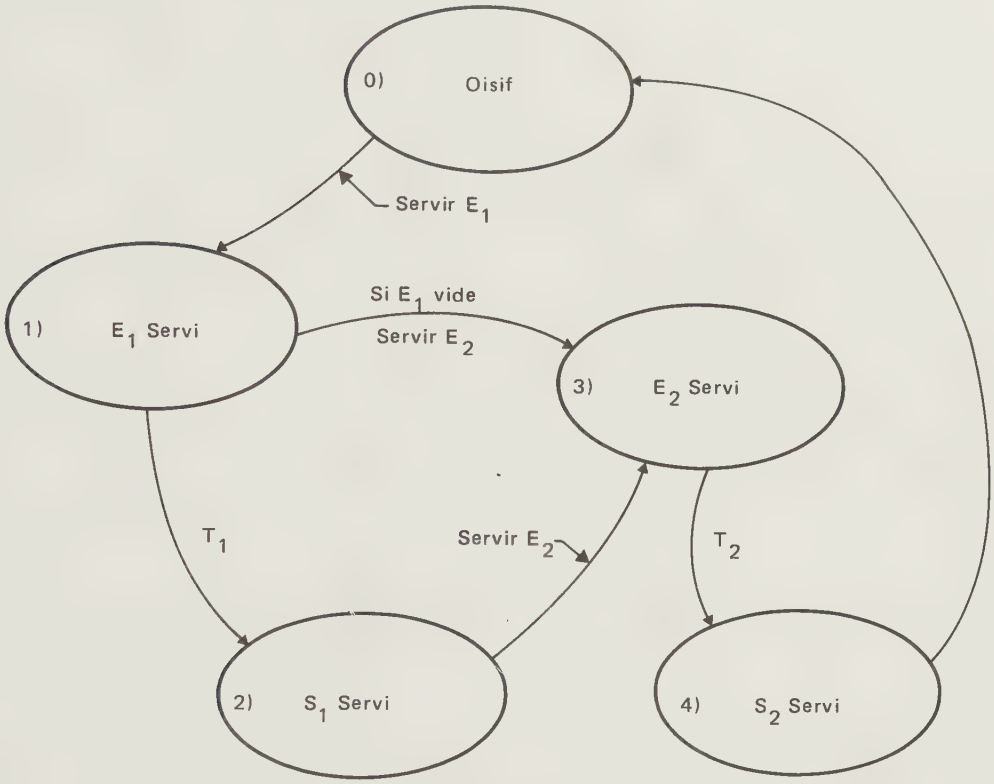
En définitive, l'algorithme de gestion des échanges peut être représenté au niveau macroscopique par un diagramme d'état (voir figure page suivante).

Ce diagramme fait apparaître cinq états. Les états sont représentés par des cercles et les actions ou processus élémentaires par des flèches. La tâche T_1 est le processus élémentaire qui consiste à prendre un mot dans la file d'échange entre le prépocesseur et l'unité de traitement microprogrammée, à le traiter (fonction de filtrage et de décision), puis à fournir une information sous forme d'un mot envoyé à S_1 à l'unité de traitement. La tâche T_2 consiste à prendre un ordre sur E_2 et à le répercuter sur S_2.

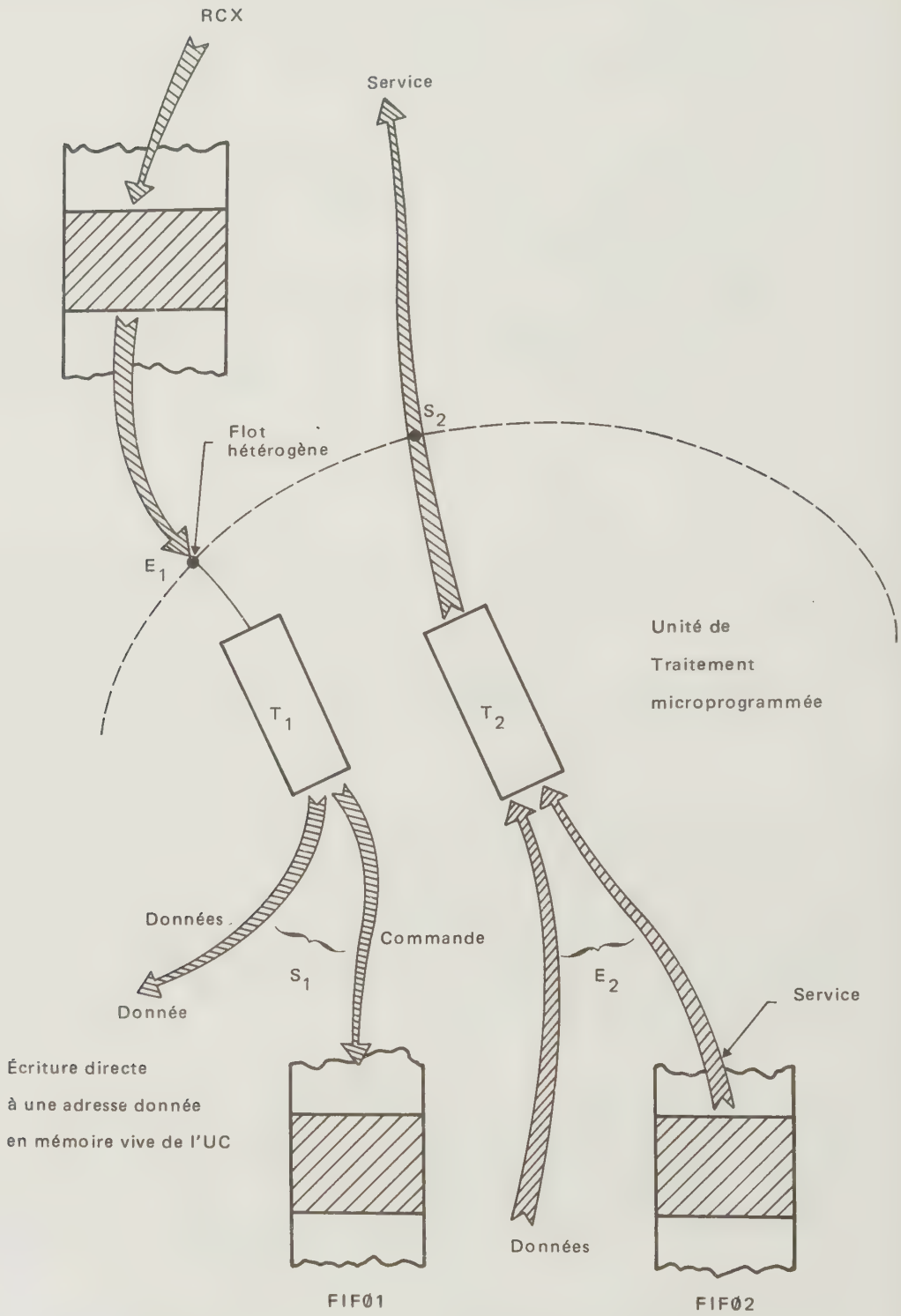
Dans ce paragraphe, on a tenté de définir un ensemble de règles qui ont permis de définir les mécanismes logiciels, mis en jeu dès l'UMS; ces règles ne dépendent uniquement que de la structure physique du système, mais il existe une autre catégorie de contraintes qui dépendent de la nature du flot de données traitées.

En effet, selon les schémas de signalisation adoptés, des contraintes de temps sont associées à chacune des phases de l'établissement ou de la rupture des communications, ces contraintes ne vont pas affecter la structure du logiciel, mais influeront sur l'optimisation des processus prépocesseur, de la programmation des tâches T_1, T_2 et de l'unité de traitement.

Diagramme d'état du processus de gestion des échanges :



3.1 - Les chemins des données



Jusqu'à présent, nous avons considéré les mécanismes d'échange entre les différents organes sans faire d'hypothèses sur la nature des informations échangées, mais en réalité, l'accès à ces points d'échange dépend de la nature des données échangées.

Les échanges entre l'UMS et l'extérieur portent sur trois catégories de données :

. Des commandes (décrochage d'un abonné, changement d'état d'une voie ...) qui sont relatives à une voie du multiplex entrant,

. Des services (commandes relatives aux organes de commande, aux RCX) ne concernant pas des opérations sur une voie,

. Des données transmises de façon transparente du RCX vers l'unité de traitement (caractères de sélection reçus sur une voie) ou de l'unité de traitement vers le RCX (message émis vers un abonné).

Suivant la nature des données transmises, le cheminement est quelque peu différent. En règle générale, les services et les commandes sont échangés au moyen de file d'attente de type FIFØ (First in First out) dont le mécanisme a déjà été expliqué, et les données transmises de façon transparente sont lues ou écrites directement dans la mémoire vive de l'unité de traitement.

Nous pouvons maintenant définir de façon plus précise le cheminement des données :

- La voie montante :

Le traitement effectué par le prépocesseur n'est pas suffisant (il s'agit avant tout d'un filtrage) pour lui permettre de gérer plusieurs points d'accès à l'unité de traitement microprogrammée. E_1 est donc un point unique d'accès, c'est-à-dire, que le mot lu dans la file d'attente par le microprocesseur comprendra toujours un caractère reçu sur une voie, un identificateur de cette voie et une indication sur le traitement en cours à initialiser.

Au niveau des échanges entre l'UMS et l'unité de traitement, le point de sortie S_1 comporte deux voies d'accès (voir le schéma ci-dessus) ; le programme d'analyse correspondant à la tâche T_1 peut conduire soit à transmettre à l'unité de traitement le caractère reçu sur la voie considérée (ou un groupe de caractères puisque le microprocesseur peut mémoriser quelques caractères dans un espace mémoire au microenregistreur associé à une voie), soit lui envoyer une commande destinée à activer une tâche de plus haut niveau concernant cette voie.

Les commandes envoyées à l'unité de traitement sont transmises au moyen d'une file d'attente (FIFØ1).

Les données sont inscrites directement dans la mémoire vive de l'unité de traitement.

A cette fin, l'unité de traitement a préalablement fourni au microprocesseur au moyen d'un message de service transmis par E_2 l'adresse d'une zone mémoire ou enregistreur allouée à cette voie pour la durée de l'établissement d'une communication, et

destinée à recevoir les caractères de sélection de l'abonné trafiquant sur cette voie.

- La voie descendante :

On a déjà dit que c'était une voie de service pour laquelle transitaient les ordres élaborés au niveau de l'unité de traitement et destinés au RCX ; pour être plus précis, il faut y ajouter une voie de transfert de message.

En effet, parmi les services offerts par le commutateur, figurent la transmission différée et la diffusion de messages qui nécessitent la mémorisation au niveau de l'UT de chaînes de caractères reçus sur une voie, ces caractères étant reçus par l'UT sur la voie de données associée à S_1.

La procédure d'utilisation de la voie de données associée à E_2 est simple, par la FIFØ2 (voir schéma ci-dessus) l'UT avertit le prépocesseur qu'un message résidant en mémoire à une adresse donnée doit être envoyé à un abonné ; le microprocesseur lit ensuite ce message caractère par caractère directement en mémoire et l'envoie au RCX sur la voie adéquate.

La voie de service transite par FIFØ2, chaque service est formulé dans un message comportant une commande codée et des indications nécessaires à l'exécution de ce service (adresse de voie, adresse en mémoire ...).

II.4 - Les trois composantes

Nous avons jusqu'à présent considéré la fonction unité d'échange de l'UMS comme un tout ; il est cependant souhaitable d'indiquer brièvement quelle est la contribution de chacun des maillons à la réalisation de cette fonction.

4.1 - Le prépocesseur

C'est une table de décision ; il reçoit à chaque intervalle de temps un ensemble d'informations relatif à une voie et comprenant :

- Un identificateur de la voie : ou adresse de voie,
- Le caractère reçu sur la voie,
- Un mot de commande décrivant l'état de cette voie, (libre, indisponible, en faute, en communication ...).

Il dispose ensuite de 976 nanosecondes pour déterminer si la voie est dans un état permanent qui ne nécessite pas l'intervention de l'organe de commande; si elle est déjà en transaction avec l'organe de commande,

dans ce cas, le prépocesseur indique à l'échelon supérieur la phase de traitement à entamer ou bien si par suite d'un changement d'état, il faut initialiser un traitement. Dans ce cas, le prépocesseur donne à l'unité de traitement microprogrammée l'ordre d'entamer une transaction avec l'abonné trafiquant sur cette voie. D'un point de vue logiciel, le processus se borne à rassembler les données à transmettre au microprocesseur et à lancer un ordre d'écriture en file d'attente, la gestion de celle-ci étant, on l'a vu, réalisée en logique câblée.

Pour la transmission dans le sens UMS vers réseau de connexion, le prépocesseur est parfaitement transparent aux données et ne fait que transférer un mot d'information que lui envoie l'unité de traitement microprogrammée dans son registre d'interface.

4.2 - L'unité de traitement microprogrammée

On a vu quel rôle prépondérant elle jouait dans la gestion des échanges entre le réseau de connexion et l'unité de traitement.

D'un point de vue fonctionnel, elle sert à marquer les voies en signalisation, c'est-à-dire qu'elle peut dans sa mémoire de travail réserver pour chacune de ces voies un petit espace - ou microenregistreur - dans lequel elle inscrit des indications concernant l'état "instantané" de la voie (adresse de la voie, dernier caractère reçu, ...). On peut entendre le terme d'état instantané par rapport à la représentation plus grossière qu'en a l'unité de traitement qui ne distingue que parmi quelques états : ligne libre, occupé, en défaut,... alors que l'UMS doit reconnaître pour en assurer la gestion plusieurs états transitoires d'une voie lors d'un changement d'état. Le passage de l'état de ligne libre à l'état d'abonné en communication sera décomposé au niveau de l'UMS en autant d'états intermédiaires qu'il y aura d'étapes dans la procédure de signalisation choisie.

Ainsi, l'unité de traitement n'aura pas à savoir exactement à quelle phase de traitement correspond la tâche qu'elle doit exécuter, il lui suffit de suivre les directives de l'UMS qu'elle suive le "film" exact de l'établissement et de la rupture des communications, ainsi que des opérations annexes de télémaintenance.

Ainsi, on peut considérer le schéma suivant :

l'UMS traitant la phase n d'une procédure de signalisation g sur une voie donnée j, va demander à l'UT d'exécuter une tâche et pendant ce temps mettra la voie j dans un état semi-permanent et attendra le compte rendu d'exécution de la tâche qui lui parviendra de l'UT sous forme d'un message de service qui lui permettra de passer à la phase n 1 de la procédure et éventuellement à mo-

difier l'état de la voie.

Pour réaliser ces différentes fonctions, l'unité de traitement microprogrammée doit disposer d'un ensemble de processus élémentaires, telles les primitives de gestion de l'espace mémoire de travail ou les primitives de gestion des entrées-sorties et de traitement de chaîne de caractères.

Ces primitives sont enchaînées les unes aux autres par un programme moniteur qui les ordonne de façon à créer les tâches que l'on a définies au paragraphe précédent. L'ensemble de ces mécanismes sera décrit au chapitre IV de ce document.

4.3 - L'unité de traitement

L'unité de traitement présente les mêmes caractéristiques du point de vue logique que l'unité de traitement microprogrammée en ce sens que la notion d'interruption a été supprimée dans la mesure du possible (toutes les voies ont même priorité, donc tous les traitements concernant ces voies doivent avoir la même priorité et il n'y a aucune raison pour interrompre un traitement par un autre de même niveau). On y trouve également la notion de tâche, c'est-à-dire, de processus élémentaires ou de chaines de processus élémentaires concourant à la réalisation d'une opération concernant une voie.

L'étude détaillée de ce logiciel n'entre pas dans le cadre du présent document. Vue de l'UMS, l'unité de traitement apparaît comme un processus unique, qui prend ses informations du point d'entrée S_1, qui élabore une décision qu'il retransmet ensuite à l'UMS par le point E_2.

Ce processus dispose d'une masse d'informations suffisante pour la gestion des communications sous forme de tables d'abonnés dans lesquelles on trouve l'état des lignes et des circuits, des classes de services et les discriminations de chaque abonné, et qui permettent également d'établir une relation entre numéro d'annuaire et adresse de la voie sur laquelle trafique l'abonné pour une communication donnée, de tables de routage qui permettent lors de l'établissement d'une communication, de déterminer compte tenu du numéro d'annuaire de l'abonné demandé, l'adresse de la voie qui lui est attribuée ou bien de trouver un circuit libre à destination du centre de rattachement de cet abonné.

A la différence de l'UMS, dont les tâches ne s'enchaînent qu'en fonction d'un contexte bien précis, l'unité de traitement a une certaine notion du temps, c'est-à-dire qu'elle dispose d'une horloge qui lui permet d'activer des tâches à des instants précis. Cette possibilité, qui est une entorse au principe que l'on s'était imposé, de supprimer la notion d'interruption,

joue un grand rôle, à la fois dans la gestion des temporisations associées à chacune des phases de la procédure de signalisation et dans la maintenance automatique du système.

En effet, l'unité de traitement peut associer à chaque tâche qu'elle entreprend ou à chaque évènement qu'elle gère, une limite de temps d'exécution. Si, à l'expiration de ce laps de temps, l'évènement n'a pas eu lieu, ou si la tâche initiale n'est pas terminée, alors le système est susceptible d'être défectueux et un processus de défense est activé.

Le fait que l'unité de taitement dispose d'une puissance de traitement supérieure à celle de l'UMS, qu'elle ait accès à une grande masse d'informations sur l'état du réseau et du commutateur lui-même, et enfin qu'elle ait la notion du temps, impose une contrainte supplémentaire pour la réalisation du logiciel de l'UMS.

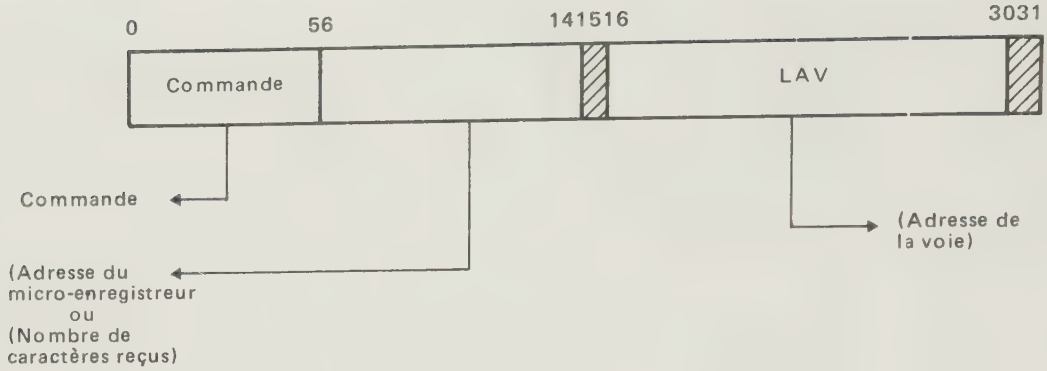
R_3 : L'unité de traitement a le pouvoir de décision final, c'est-à-dire qu'aucune action ne peut être entreprise au niveau du RCX, envoi d'un octet sur une voie donnée ou changement d'état d'une voie, sans ordre préalable de l'unité de traitement.

Cette règle est fondamentale et constitue le souci premier du programmeur de l'UMS.

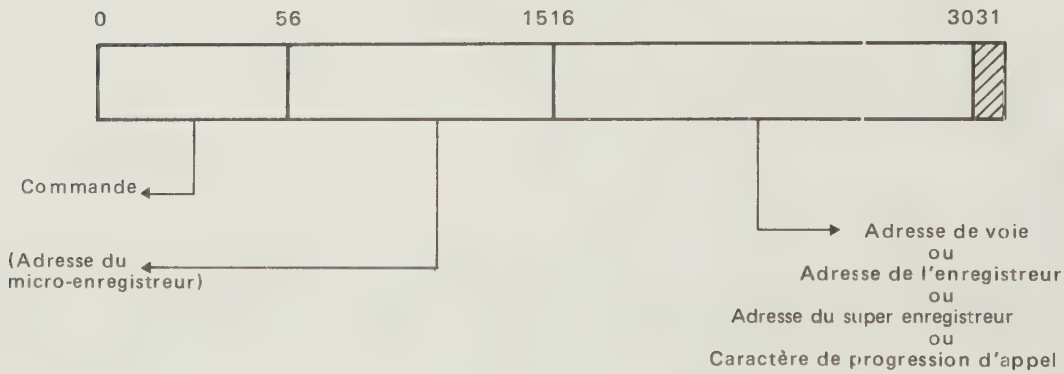
Un des points importants qui ont également guidé le choix d'une structure de programmation pour l'UMS est que les logiciels de l'UMS et de l'UT doivent être aussi indépendants que possible, c'est-à-dire, qu'un processus actif sur l'une des deux machines n'aura pas besoin de connaître la structure du processus concurrent actif sur l'autre machine. Cela sous-entend l'existence d'un langage ou protocole de gestion des échanges entre les deux unités.

Les messages qui transitent par les files FIFØ1 et FIFØ2 (points S_1 et E_2) ont un format standard et sont codés sur 32 e.b. Les commandes et demandes de service échangées sont généralement relatives à une opération concernant une voie de transmission, le message doit donc comporter un identificateur de la voie à traiter, l'UMS le fait en donnant à l'UC une adresse de voie (LAV_{0-15}) et l'UC donne à l'UMS l'adresse au microenregistreur servant à marquer la voie que l'on va traiter, ce qui est suffisant puisque l'UMS attribue un microenregistreur à chaque voie en transaction avec le commutateur qu'elle libère en fin d'opération. Le message comporte également un code d'ordre sur 6 e.b., ce qui permet à chacun des deux organes d'échanger 64 types de messages différents dans les deux sens, et des informations diverses nécessaires à l'exécution de la tâche commandée.

a) Sens UMS vers UC



b) Sens UC vers UMS



Le format du message échangé est décrit ci-dessus ; le détail des commandes et leur codage est donné en annexe (I) et il en sera fait mention ultérieurement.

En définitive, cette procédure permet de rendre les deux logiciels parfaitement indépendants du point de vue structure, même lorsque l'on considére le cas où l'unité de traitement microprogrammée doit lire ou écrire des données dans la mémoire de l'UT, car, alors, l'échange a été initialisé par un message donnant l'adresse précise de lecture ou d'écriture que le microprocesseur "oubliera" aussitôt l'opération terminée.

II.5 - La gestion des voies logiques

Pour comprendre comment s'enchaînent les différents processus on peut suivre schématiquement l'établissement d'une communication entre deux abonnés. Nous prendrons le cas d'une communication simple, c'est-à-dire, entre deux abonnés desservis par le même commutateur. Pour initialiser une communication, les deux abonnés doivent suivre une procédure dont le diagramme d'état est donné en annexe (II). Nous en rappelons ci-dessous les grands principes :

L'équipement terminal d'un abonné (A) ou (B) peut être considéré comme un automate. Prenons le cas où (A) veut entrer en communication avec (B), (A) serait demandeur et (B) demandé.

L'établissement de la communication entre demandeur et demandé doit suivre le schéma suivant :

1	→	(A) est dans l'état de ligne libre.
2	→	(A) formule une demande d'appel en modifiant certains paramètres d'état de sa ligne.
	←	Le centre de rattachement de (A) lui envoie après prise en compte de sa demande d'appel une invitation à numéroter
3	→	(A) émet successivement ses caractères de sélection (numéro d'annuaire de (B)) qui sont reçus et mémorisés par le commutateur.
4	→	Lorsque tous ses caractères de sélection ont été envoyés, (A) envoie une indication de fin de sélection, qui doit être immédiatement interprétée par le commutateur.
5	←	Le commutateur laisse (A) en attente, détermine la voie attribuée à (B) et envoie vers (B) une indication d'arrivée d'appel.
6	→	(B) doit alors émettre un signal de confirmation indiquant qu'il est prêt à accepter l'appel ar-

	rivant
7 ←	Dès la réception du signal de confirmation, le commutateur envoie vers (A) le demandeur, un caractère de progression d'appel, ce caractère est utilisé pour indiquer au demandeur si sa communication est possible ou non et, alors, pour quelle raison.
8	Dans le cas où la communication est possible, les deux abonnés (A) et (B) sont mis en présence dès l'envoi du caractère signal de progression d'appel au demandeur.

On a donné ci-dessus le cheminement de la procédure vue de l'extérieur, c'est-à-dire, du côté abonné ; il est intéressant d'en considérer les huit étapes du point de vue de l'enchaînement des mécanismes élémentaires au niveau de l'organe de commande du commutateur.

1	(A) est dans un état permanent (ligne libre) aucune action n'est entreprise au niveau de l'organe de commande (fonction filtre du prépocesseur).
2	(A) voulant initialiser un appel, le prépocesseur détecte une modification de certains paramètres de sa ligne et met dans sa file un message à destination de l'UT microprogrammée comprenant l'adresse de la voie de (A) et l'indication "détection d'appel". Le programme moniteur du microprocesseur a pour tâche, on l'a vu, de vider cette file et il finira par recevoir ce message du prépocesseur. Ce message décodé (tâche T_1), il marquera la voie de (A) en lui attribuant un microenregistreur dans sa mémoire de travail et il enverra par S_1 dans FIFØ1 un message du type "décro-

	chage". Lorsqu'il lira ce message, le programme moniteur de l'UT activera des tâches qui vérifieront la validité de la demande de (A) qui réserveront des ressources - enregistreurs - qui permettront la gestion de la procédure et qui, en définitive, inscriront dans FIFØ2, un message (invitation à numéroter) à destination du microprocesseur donnant à celui-ci l'autorisation de poursuivre la procédure et lui indiquant l'adresse de la zone mémoire réservée à (A) pour la durée de la communication. Celui-ci enverra sur la voie de (A) par l'intermédiaire du prépocesseur, le caractère (invitation à numéroter) en même temps qu'il procèdera au changement d'état de la voie de (A), par modification de son mot de commande, en le plaçant dans un état de signalisation.
3	(A) émet successivement tous ses caractères de sélection, ils sont reçus par l'UMS (prépocesseur → file d'attente → unité de traitement microprogrammée), groupés par deux pour limiter les échanges avec l'UT et inscrits directement par S_1 dans la mémoire vive de l'UT (dans l'enregistreur associé à (A)).
4	Le prépocesseur reconnaît le caractère de fin de sélection de (A) et fournit à l'unité de traitement microprogrammée le message de fonction correspondant. Celui-ci en avertit l'UT par un message transitant par la file FIFØ1 (comprenant en outre le nombre de caractères envoyés par (A)), et fait passer (A) dans un état d'attente par modification de son mot de commande.

5	Lorsqu'il analyse le message de fin de sélection élaboré par l'UMS, le programme moniteur de l'UT trouve dans l'enregistreur de (A) l'ensemble de ses caractères de sélection et active alors un processus complexe qui, à partir des caractères de sélection émis par (A), détermine le service exact demandé par (A) (connexion simple, diffusion ...), identifie le demandé (B), vérifie si (B) est dans un état tel qu'il puisse être appelé (condition (B) libre) et extrait son adresse de voie. Alors commence la procédure par l'autre extrêmité, l'UT met dans FIFØ2 un message, lui donnant l'ordre d'envoyer à (B) une indication d'arrivée d'appel qui est interprétée par l'UMS comme une autorisation de modifier l'état de la ligne de (B), ce qu'elle fait en avertissant (B).
6	(B) est dans un état d'attente jusqu'à ce qu'il émette un caractère de confirmation. Cet accusé de réception est reconnu par le prépocesseur qui en avise le microprocesseur par un message adéquat. Ce message, reçu sur E_1 par le microprocesseur est réémis sur S_1 de façon que l'UT connaisse la réaction de (B).
7	Dès lors, l'UT sait que la communication est possible et envoie à l'UMS l'ordre d'émettre vers (A) un caractère de progression d'appel (succès ou échec).
8	Après l'envoi du caractère de progression d'appel, les deux abonnés (A) et (B) sont placés dans un état d'attente de connexion et l'UT peut donner l'ordre de connexion à l'UMS. Cet ordre parvient au microprocesseur sous forme

	d'un message lu sur E_2. A ce moment, le microprocesseur dispose des deux adresses de voie, la connexion s'effectue en inscrivant successivement dans le mot de commande de (A) l'adresse de voie de (B) et dans le mot de commande de (B) l'adresse de voie de (A), et en les mettant tous deux en état de communication. Un accusé de mise en connexion est ensuite envoyé à l'unité de traitement.

Ce qui précède n'est qu'un exemple concret n'ayant pour but que de mettre en évidence le mode d'enchaînement des différents processus nécessaires à la gestion du réseau. Ce cas était celui de la communication la plus simple; pour des services plus élaborés (diffusion, transmission différée de messages, ...) les mécanismes sont les mêmes si les processus diffèrent quelque peu. Une étude exhaustive de ces processus sera faite au dernier chapitre de ce document.

III. - THEORIE DE LA TABLE DE DECISION

III.1. - Analyse fonctionnelle

1.1. - Le préprocesseur comme automate programmable :

Dans le schéma général de l'organe de commande le préprocesseur joue, on l'a vu un triple rôle :

1) - Adaptation d'interface par transformation de la fonction synchrone du RCX en une fonction asynchrone.

2) - Prétraitement des informations reçues du RCX.

3) - Filtrage de l'information reçue du RCX.

Le premier point peut être considéré comme une conséquence de la structure physique de cet organe et sera décrit dans le document consacré à sa structuration (chapitre III), dans ce chapitre seront plutôt développés les points (2) et (3).

Au préprocesseur ont été assignées deux tâches essentielles :

- la supervision de toutes les voies du multiplex entrant.
- la télémaintenance du système.

Le problème essentiel est d'une part de définir ce qu'on entend par supervision et télémaintenance en terme de fonction et d'autre part d'imposer des limites (inférieures et supérieures) au degré de sophistication de ces deux tâches puisqu'on le verra ce facteur a une influence prépondérante sur la charge des autres maillons de la chaine des données et principalement sur celle de l'UMS.

Une limite inférieure peut être aisément trouvée, elle est imposée par la capacité de traitement du microprocesseur, on peut définir ainsi un filtre minimum. La limite supérieure est une notion plus difficile à cerner parce qu'un peu subjective. En effet la raison qui a conduit à introduire le préprocesseur dans la chaine des données était la contrainte de temps, il fallait disposer d'un organe capable d'absorber un débit d'information équivalent à un mot de 44 e.b. toutes les 976 ns soit 45.056 Mb/s, et puisqu'aucune structure de calculateur universel ne pouvait convenir il a fallu définir une fonction et réaliser cette fonction en logique câblée qui pouvait seule satisfaire aux contraintes extérieures.

Donc puisque le préprocesseur se présente du point de vue fonctionnel comme un processus câblé et que contrairement à ce qui se passe dans la majorité des applications informatiques c'est le logiciel qui influence la structure du matériel, on pouvait en théorie rendre ce processus aussi sophistiqué que possible en compliquant sa réalisation.

Ces considérations et en particulier la dernière, nous ont conduit à dégager plusieurs conclusions qui ont été les bases de la structure matérielle et logicielle du préprocesseur :

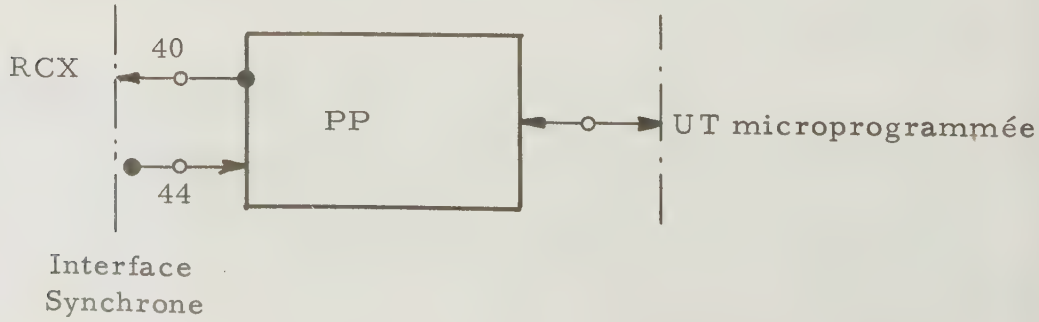
1) - Le préprocesseur doit être un organe relativement simple à réaliser, il n'existe qu'en tant qu'aide aux organes de niveau supérieur et en aucun cas ne doit se substituer à eux. Ceci en particulier contribue à définir ce qu'on entend par limite supérieure du degré de sophistication de la fonction préprocesseur.

2) - Le préprocesseur est un organe câblé ; si l'on accepte q le processus qu'il supporte soit câblé, on voit disparaître du coup toute possibilité d'évolution, toute souplesse de mise au point. Il importe donc que le produit final ne soit pas une simple fonction de logique combinatoire mai puisse être en partie du moins programmable.

3) - Il ne faut pas perdre de vue que ce qui est réalisé s'appe une maquette, c'est-à-dire un dispositif expérimental servant à démontrer la faisabilité d'une nouvelle techniqu de commutation et d'instrument de mesure de certains paramètres jusqu'alors mal connus. L'un de ces paramèt est la charge de chacun des maillons de la chaine en particulier de l'unité de traitement microprogrammée do les performances sont encore mal évaluées. Notre objec est de réaliser une sorte de lissage du taux d'utilisatio de chacune des composantes et pour ce faire il est néces re que l'on puisse à tout moment modifier le comportem de l'un quelconque des processus en jeu, ceci implique qu non seulement le préprocesseur doit être un organe programmable mais également reprogrammable.

La considération de ces trois points particuliers nous a en définitive conduit à concevoir le préprocesseur comme un automate programm Sa structure définitive est décrite dans le document concernant le matériel, elle découle de l'analyse fonctionnelle exposée ci-après :

1.2. - La fonction filtre du préprocesseur :



La première fonction du préprocesseur est de superviser et de filtrer toutes les voies du multiplex entrant, ce multiplex a trois caractéristiques essentielles :

1) - Il est synchrone, l'intervalle de temps élémentaire à son niveau a une durée de 976 nanosecondes.

2) - Il est hétérogène c'est-à-dire qu'il comporte des voies de natures différentes.

- 2400 b/s s
- 9600 b/s (ultérieurement)
- 50 bauds télégraphique

Les voies à 2400 et 9600 b/s sont synchrones.

Les liaisons à 50 Bauds sont arythmiques.

3) - Chacune des voies du multiplex respecte un schéma de signalisation c'est-à-dire que le comportement d'une voie n'est pas aléatoire :

- une voie est caractérisée à un instant donné par un état particulier.
- elle ne pourra changer d'état que si certaines actions précises sont entreprises.
- à partir d'un état particulier seul un nombre restreint d'autres états sont accessibles et donc seul un petit nombre d'actions différentes sont admises.

On peut essayer de développer chacun des trois points précédents pour définir la fonction préprocesseur.

1) - Le multiplex entrant, l'interface UMS-RCX :

Le préprocesseur présente avec le RCX une interface synchrone, cette interface est définie de façon précise dans la note : Spécifications techniques et logiques concernant l'interface entre le réseau de connexion du commutateur Hermès et son organe de commande (3) (4), nous en rappelons ici quelques caractéristiques.

A chaque intervalle de temps le RCX envoie vers le préprocesseur un ensemble d'informations concernant une voie c'est-à-dire :

- L'adresse de la voie : LAV (15 e.b.)
- Le caractère émis par l'abonné avec son bit S : CR (9 e.b.)

- Le mot de commande de la voie : LMC (16 e.b.)
- La phase de l'octet à émettre par le préprocesseur : PHQ (2 e.b.)
- L'état du caractère reçu : ECR (1 e.b.)
- Une horloge qui valide les informations précédentes : BT (1 e.b.)

Durant le même intervalle de temps le préprocesseur doit émettre vers le réseau de connexion un mot de 40 e.b. comprenant :

- un octet de signalisation : OSE (8 e.b.)
- un mot de commande à écrire en mémoire de commande : EMC (16 b.)
- une adresse d'écriture en mémoire de commande : AMC (15 e.b.)
- une demande d'écriture en mémoire de commande : CEM (1 e.b.)

L'octet de signalisation doit être obligatoirement émis à chaque IT par le préprocesseur, il sera inscrit dans la mémoire tampon à l'adresse de la voie incidente (LAV) c'est-à-dire envoyé à l'abonné dont on vient de recevoir un caractère, toutefois si l'abonné est dans l'état de communication l'octet envoyé par l'organe de commande lui sera pas transmis, le RCX est donc capable de discerner deux états d'une voie :

- en communication: il échange simplement les informations des voies émettrice et réceptrice.
- en signalisation: il effectue simplement un bouclage de la ligne.

Les autres informations émises par le préprocesseur EMC, AMC, CEM sont facultatives, mais la seule action que peut exercer l'organe de commande sur le réseau de connexion est la modification d'un mot de la mémoire de commande, c'est pourquoi l'UMS peut à tout moment (mais jamais deux fois dans le même IT) demander (par CEM) l'écriture du mot (EMC) à l'adresse (AMC) de la mémoire de commande, cet ordre est immédiatement exécutable par le RCX.

En définitive le préprocesseur dispose au début de chaque IT, de tout un contexte définissant l'état d'une voie à cet instant précis.

Et son rôle consiste à déterminer quel traitement exacte nécessite l'état de la voie. Pour cela on lui alloue toute la durée de l'IT pendant laquelle il ne traitera qu'une voie. On retrouve ici la notion de tâche définie au chapitre précédent, en effet le processus activé en début d'IT est un processus élémentaire non interruptible qui lit une information sur un point d'entrée (l'interface préprocesseur RCX) qui la traite, la filtre et transfert éventuellement cette information au niveau supérieur. Lorsqu'il a terminé son traitement le processus se met en sommeil, il ne sera réactivé par un moyen externe qu'au début de l'IT suivant, pendant le temps de sommeil le préprocesseur est complètement inactif à la différence du microprocesseur où un processus en entraîne obligatoirement un autre, le préprocesseur ne supporte qu'un processus unique qui est incapable de boucler sur lui-même.

Pour aider à la compréhension des explications futures nous donnons ci-dessous un résumé des conventions utilisées par la représentation des informations échangées entre l'UMS et le RCX.

FORMAT ET CADRAGE DES INFORMATIONS ECHANGEES ENTRE L'UMS ET LE RCX

-:-:-:-:-:-:-

Désignation	Signification
BT	Horloge symétrique à la fréquence octet du multiplex entrant.
PHQ	Phase de l'octet à émettre, dans le cas d'une voie arythmique 50 bauds on a PHQ = 0.
ECR	ECR = "0" Cas normal c'est-à-dire qu'il n'y a pas d'alarme et que l'information reçue est significative. ECR = "1" Alarme ou caractère reçu non significatif.
CR	Caractère reçu C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 S Le mot reçu contient codé sur 9 e.b. : a) Un caractère de signalisation ou de données sur 8 e.b. (C_i | i : = 0, 7) Dans le cas d'un caractère de signalisation le code adopté est soit le code CCITT n° 5 à 7 moments plus un moment de parité soit le code CCITT n° 2 à 5 moments (voies arythmiques à 50 bauds). Pour la signalisation en code CCITT n° 2 le caractère est cadré sur les e.b. C_3 C_4 C_5 C_6 C_7, on convient alors que : (C_0 C_1 C_2) = "000" b) Un élément binaire S - Si le caractère provient du RCX synchrone c'est l'e.b. d'état. - Si le caractère provient du RCX télex c'est l'e.b X (de polarité). L'e.b. S est également utilisé pour la signalisation des alarmes.

LAV	Ligne d'adresse de voie 0 14 A_0 A_1 - A_{13} T Ce mot contient, codée sur 15 e.b. l'adresse de voie de l'octet entrant. Il est divisé en deux champs : a) Un e.b. (T) indiquant le type du RCX d'où provient l'octet entrant, avec la convention. (T) = "0" RCX synchrone (T) = "1" RCX télex b) Un champ servant à coder l'adresse de la voie dans le multiplex propre au RCX d'où elle est issue (Ai | i : = 1,13). Les poids faibles de l'adresse sont cadrés à droite.
LMC	Lecture mémoire de commande : (16 e.b.) C (a) A T 1 (b) S_0 S_1 S_2 S_3 S_4 0 C'est le mot de commande. Il est divisé en deux champs, l'e.b. C et un champ d'information de 15 e.b., C est l'e.b. de communication. a) C = "1" : la voie est en communication, le RCX trouvera dans le champ d'information l'adresse d'écriture de l'octet incident en mémoire tampon avec les mêmes conventions que ci-dessus. T = "0" RCX synchrone T = "1" RCX télex
OSE	Octet de signalisation émis : (8 e.b.) F 0_0 0_1 0_2 0_3 0_4 0_5 S_X

Deux cas sont à considérés :

a) Si la voie est une voie synchrone, l'octet est un octet de quadrioctet, on a alors :

- F = "0"
- (0i | i : = 0, 5) contient l'information utile
- S_x peut être un e.b. S_A , S_B, S_C, S_D

- Si $S_x \neq S_A$
  Alors S_x = "ON" (= "0")
- Si $S_x = S_A$
  Alors cet octet peut véhiculer des informations utiles à la maintenance.

b) Si la voie est une voie arythmique 50 Bauds les e.b. F et S n'ont plus de signification du point de vue de l'information émise, on conservera la même convention :

- F = "0"
- (0i | i : = 0, 5) caractère émis (code CCITT n° 2)
- S est réservé à la maintenance.

CEM

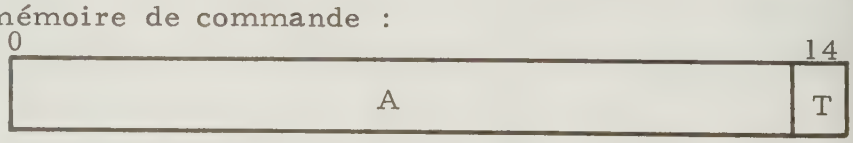
Commande écriture mémoire :

CEM = "1" Demande d'écriture du mot (EMC) à l'adresse (AMC).

CEM = "0" pas d'écriture en mémoire de commande.

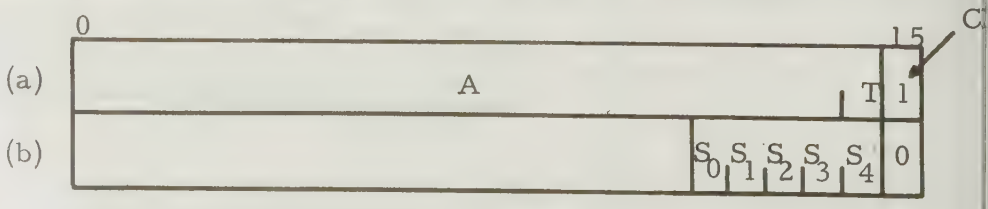
AMC

Adresse mémoire de commande :



On adopte pour la désignation de l'adresse la même convention que pour LAV.

EMC



C = "1" abonné en communication

C = "0" abonné dans un autre état

La convention est la même que pour LMC.

2) - Le préprocesseur doit traiter un multiplex hétérogène puisque toutes les voies n'ont ni le même débit (50 bauds, 2400 b/s et 9600 b/s) ni la même nature (synchrone ou asynchrone) c'est à dire qu'on doit le concevoir comme un automate capable de reconnaître instantanément la nature de la voie qu'il considère et de lui appliquer le traitement adéquat. Le paramètre dont il dispose pour identifier la nature de la voie qu'il doit considérer est dans le mot LAV l'e.b. 14 (T), dans l'état actuel des choses, un seul élément binaire suffit puisqu'il n'y a que deux procédures possibles : synchrone pour T = "0", asynchrone pour T = "1". Dans une phase ultérieure, au fur et à mesure que l'on élargira la gamme des services, on pourra aisément disposer d'autres éléments binaires.

On s'est imposé en définissant le préprocesseur deux contraintes logiques essentielles :

- il doit être capable de traiter n'importe quelle procédure de signalisation.
- il doit, dans une configuration donnée pouvoir accepter simultanément plusieurs types de procédures.

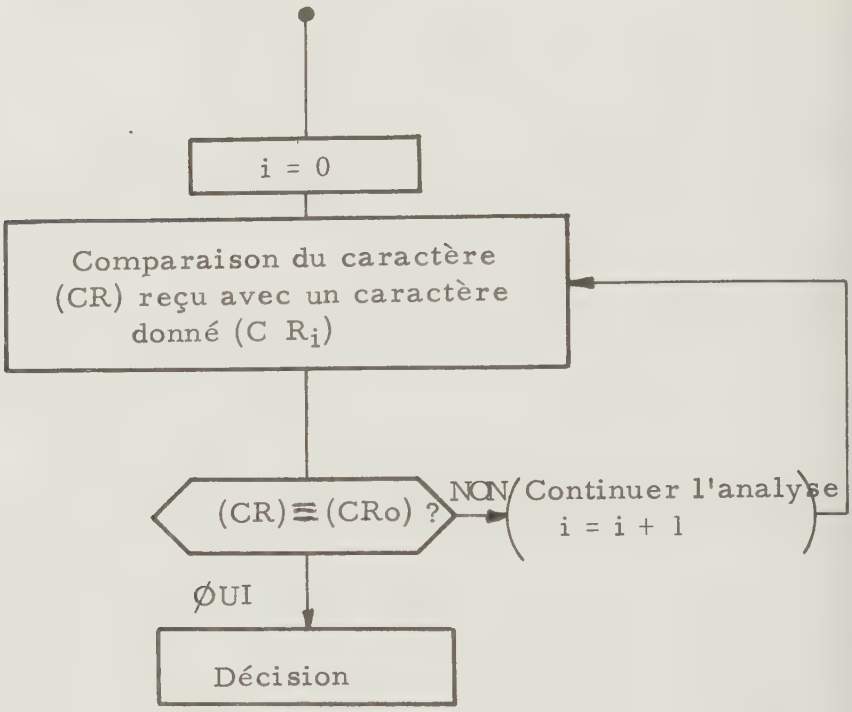
La seconde contrainte est, compte tenu de la structure choisie, la plus facile à satisfaire puisque le nombre de procédures acceptées à un instant donné par le préprocesseur ne dépend que de la taille de son "programme" et donc du dimensionnement physique des tables qui supportent ce programme.

La première contrainte est beaucoup plus fondamentale, elle nous a imposé une analyse exhaustive de tous les systèmes de signalisation de façon d'une part à dégager les principes fondamentaux de cette "linguistique" et d'autre part à s'assurer qu'il n'existait dans notre structure aucun cas d'impossibilité.

Ajoutons encore que l'analyse des quelques systèmes de signalisation existants ou à venir nous a conduit à une remarque fondamentale pour la conception du processus préprocesseur :

- quelle que soit la nature de la voie considérée la tâche d'analyse du contexte fourni par le RCX peut être décomposée en une suite de mécanismes élémentaires qui peuvent être utilisés de façon récursives pour l'exécution d'une même tâche, à la façon des boucles de programme dans un calculateur universel. En outre ces mécanismes élémentaires peuvent être utilisés pour tout type de procédure de signalisation.

Un exemple d'un de ces mécanismes élémentaires est décrit dans l'organigramme ci-dessous. Il consiste en l'analyse du caractère reçu sur une voie. Le mécanisme est initialisé pour chaque nouvelle voie et il compare successivement le caractère reçu (CR) à une suite de caractères (paramètres) le processus boucle jusqu'à ce que le caractère reçu soit identifié à l'un des caractères présentés, dans ce cas une décision est prise et un autre mécanisme élémentaire est initialisé.



3) - Chacune des voies incidentes doit respecter une procédure de signalisation, c'est-à-dire des règles de dialogue avec l'organe de commande pour la formulation des demandes de services et on a déjà vu que l'on voulait faire du préprocesseur un organe aussi universel que possible, cependant pour des raisons pratiques (il n'existe que deux types de voies) le préprocesseur n'est programmé que pour deux types de signalisation :

- la procédure télex [18]
- la procédure synchrone spécifique et la maquette de commutation, cette procédure est entièrement décrite dans l'annexe (II) elle est très voisine de la procédure qui sera adoptée au niveau international pour les nouveaux réseaux de données, c'est pourquoi dans la suite nous traiterons toujours ce cas en exemple.

Le diagramme d'état de cette procédure est donné dans les pages suivantes.

1.3. - Principe de fonctionnement du préprocesseur :

Le processus activé au début de chaque IT assume deux fonctions bien distinctes qui correspondent à la double nature (synchrone et asynchrone) du préprocesseur qui sont les fonctions d'analyse et de traitement.

1) - La fonction d'analyse :

La première de ces fonctions est la fonction de surveillance des voies entrantes, ce qui revient à analyser toutes les données reçues du RCX de façon à détecter les demandes de changement d'état de chaque voie.

Une voie est définie par son état, qui est l'état dans lequel l'organe de commande l'a placée c'est-à-dire qu'une voie donnée n'est pas libre de se placer d'elle même dans un état particulier. L'état présent de la voie est codé sur 5 e.b. (+ 1 e.b.) dans le mot de commande associé.

Le codage se fait en deux champs, l'e.b. C est connu à la fois du RCX et du préprocesseur, avec la convention suivante :

C = "1" la voie est en communication

C = "0" la voie n'est pas en communication

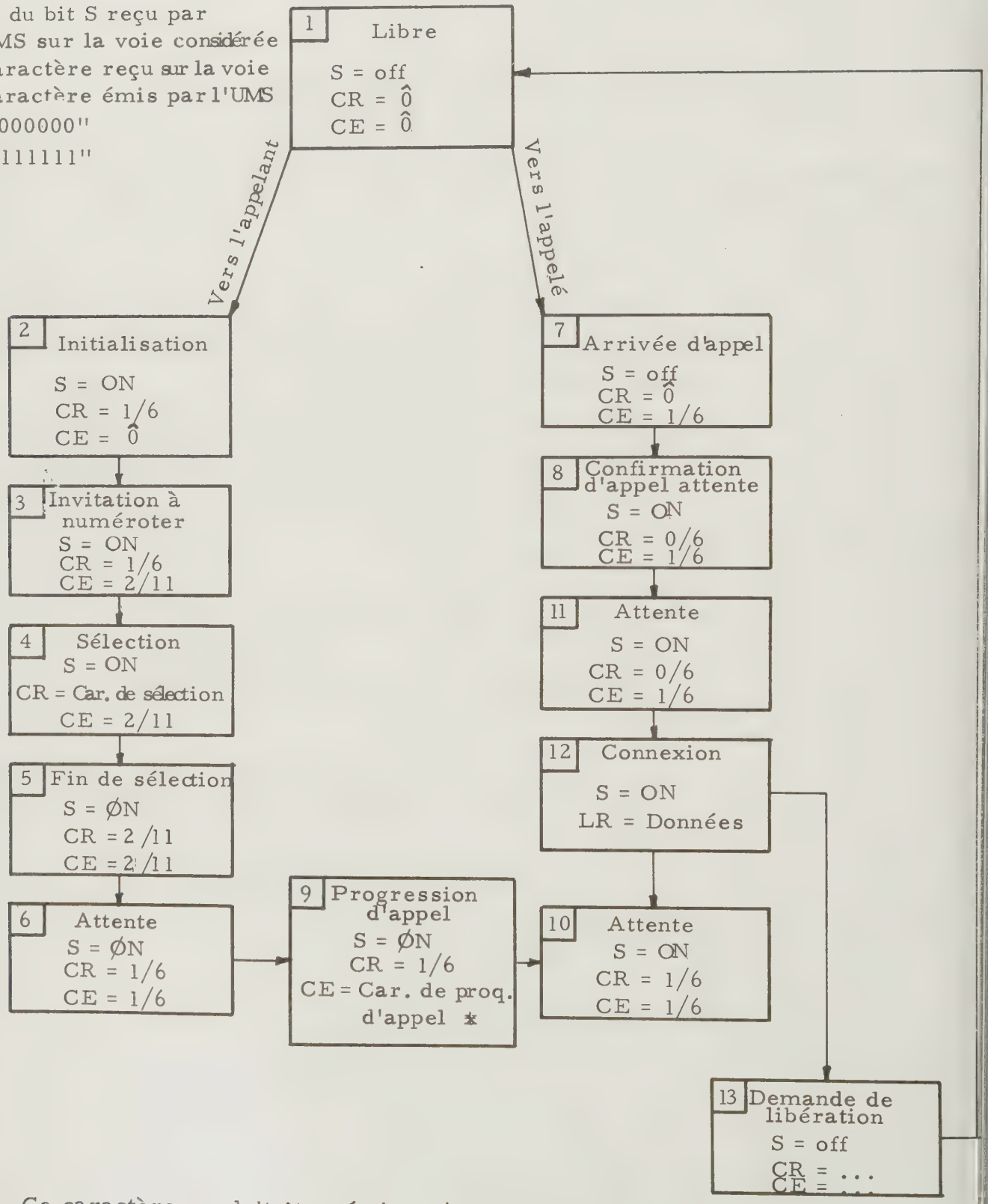
DIAGRAMME D'ETAT DU SCHEMA DE SIGNALISATION SYNCHRONE

COMMUNICATION SIMPLE

-:-:-:-:-:-

Légende :

S = état du bit S reçu par l'UMS sur la voie considérée
CR = caractère reçu sur la voie
CE = caractère émis par l'UMS
$\hat{0}$ = "00000000"
$\hat{1}$ = "11111111"



* Ce caractère ne doit être émis qu'une seule fois.

DIAGRAMME D'ETAT DE LA SIGNALISATION SYNCHRONE

APPEL DIRECT - HOT-LINE

-:-:-:-:-:-

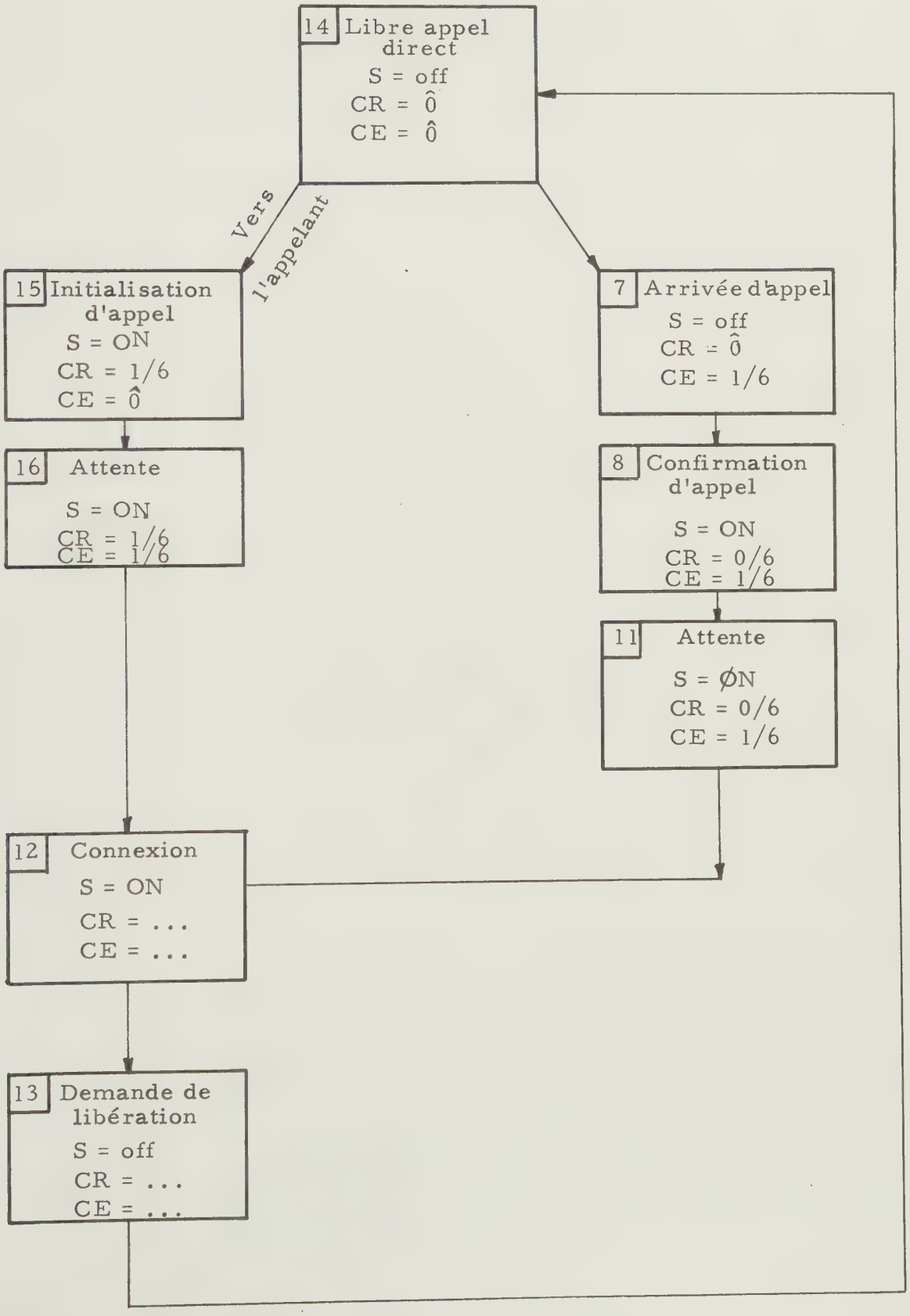
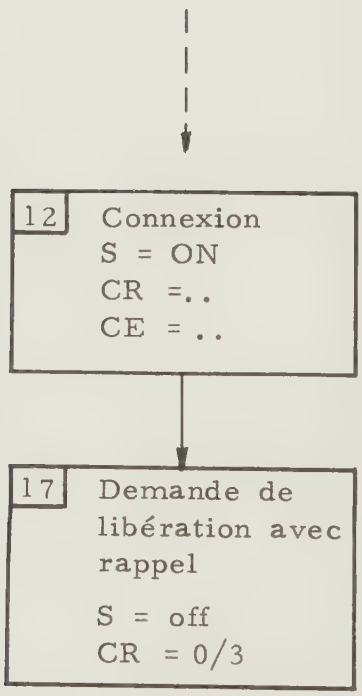


DIAGRAMME D'ETAT DE LA PROCEDURE DE SIGNALISATION SYNCHRONE

RUPTURE DE COMMUNICATION AVEC RAPPEL IMMEDIAT

-:-:-:-:-:-:-:-:-



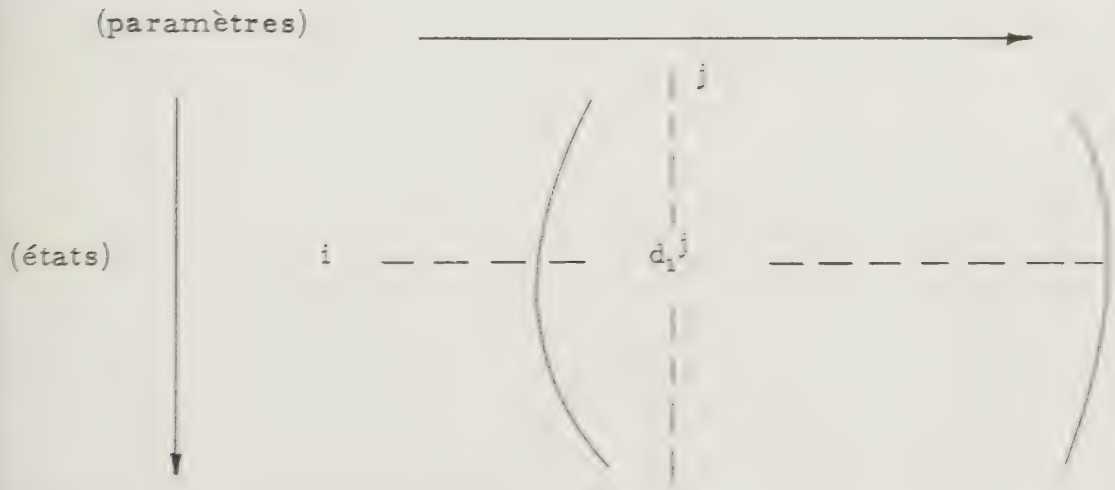
Lorsque la voie est en communication le reste du mot sert à coder l'adresse de voie du correspondant. Lorsqu'elle n'est pas en communication (C = 0) le champ (S_0 S_1 S_2 S_3 S_4) est récupéré à l'usage du préprocesseur seul pour coder d'une façon plus précise l'état de la voie. Donc on pourra distinguer $2^5 + 1 = 33$ états différents pour toutes les voies d'un même type, la distinction entre les différents types de voies étant faite par l'e.b. T de LAV.

Les paramètres que peut utiliser un abonné pour marquer une demande de changement d'état sont :

- le caractère émis CR
- le bit S associé

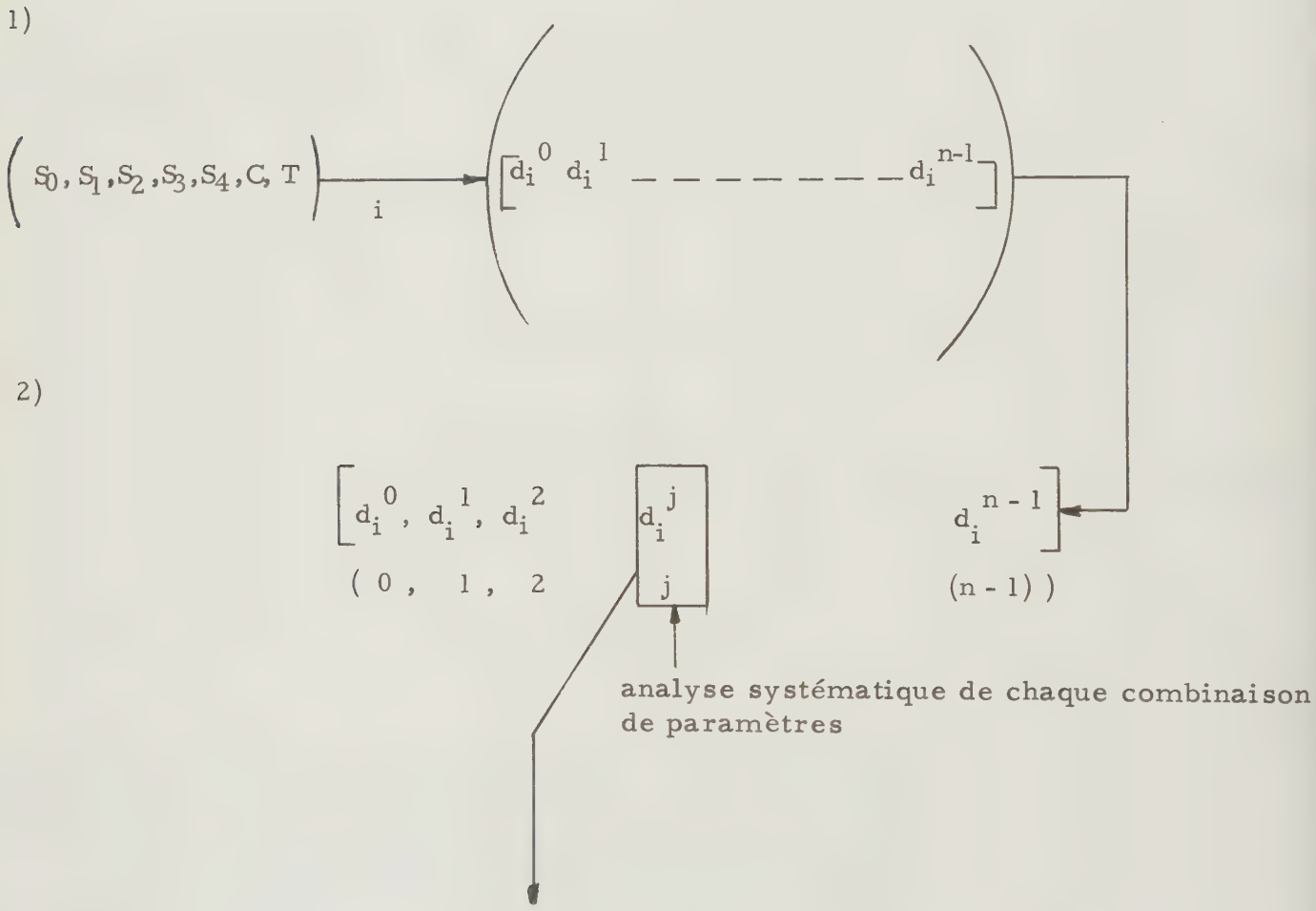
- La table de décision théorique

La partie analyse du préprocesseur fonctionne à la manière d'une table de décision, c'est-à-dire d'une matrice $d_I^{\sim}$ (voir figure ci-dessous).



dont chaque ligne est représentative d'un état et chaque colonne une combinaison des paramètres de définition de la voie (CR, S). Cette représentation n'est pas aberrante puisque l'on considère qu'une voie étant dans un état donné on ne peut exercer qu'un nombre limité d'actions pour la faire changer d'état, il y aura donc un nombre fini de combinaisons de paramètres possibles. L'élément (d_i^j) lu est une décision exploitée par la fonction traitement (qui peut être un traitement local entièrement exécuté par le préprocesseur, ou bien en ordre de lancement d'un processus du microprocesseur).

Pour analyser un contexte donné, le préprocesseur décode l'état de la voie et lit la ligne correspondante de la matrice de décision. C'est ce qu'on peut appeler l'analyse verticale. Ensuite vient la phase d'exploitation horizontale de la ligne sélectionnée. Le processus considère alors successivement chaque cas possible c'est-à-dire chaque colonne et le compare au cas réel c'est-à-dire à la combinaison de paramètres reçue du RCX. Lorsqu'il a trouvé la coïncidence le processus s'arrête et extrait l'élément de décision de la case sélectionnée (voir figure ci-dessous).



Le cas "j" a été reconnu comme conforme à la réalité on extrait la décision .

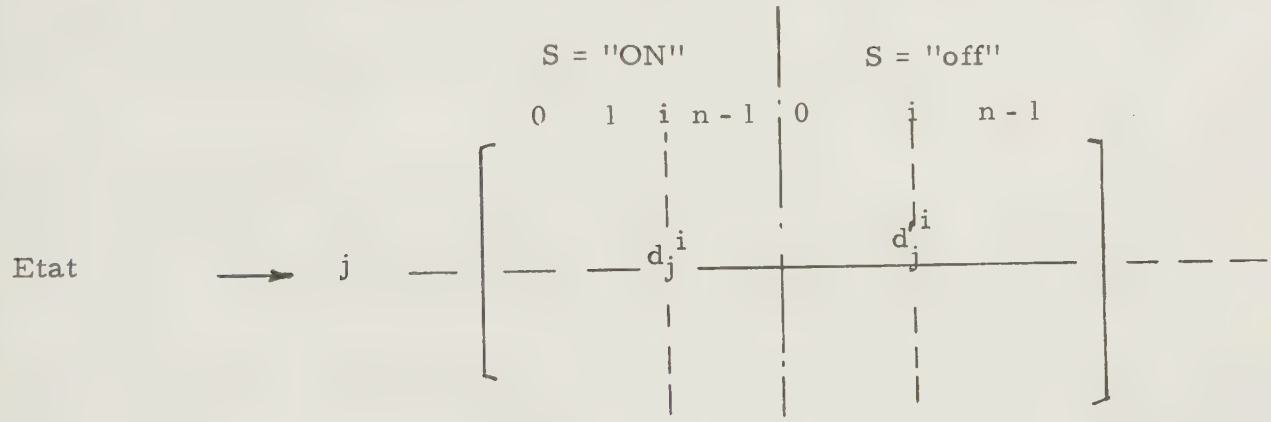
- Les problèmes de dimensionnement

Puisque nous avons réduit le problème de l'analyse du contexte à une opération sur une matrice, nous voyons apparaître immédiatement le problème du dimensionnement de cette matrice.

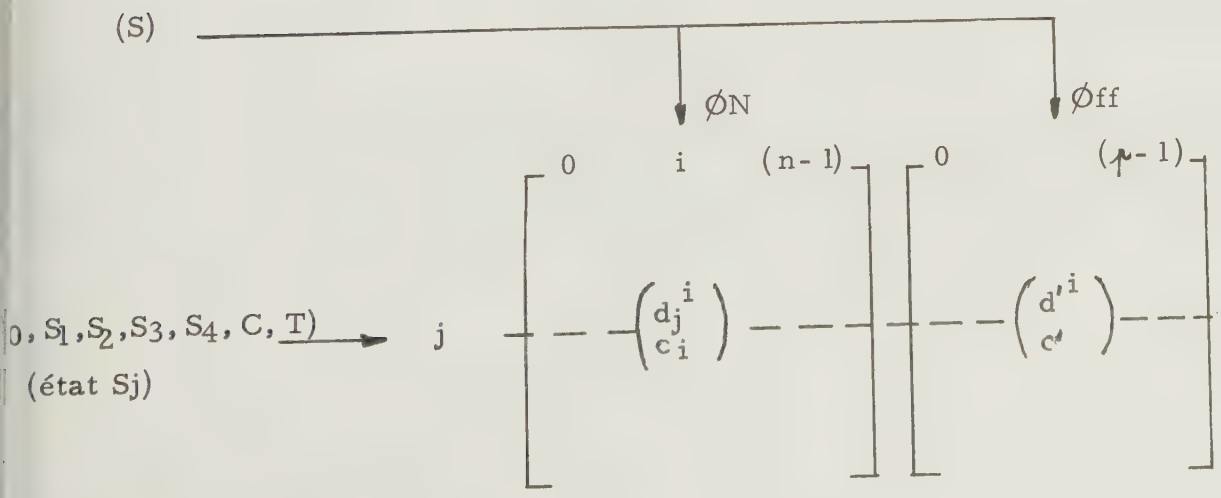
- le nombre de lignes est fixé à priori, il doit être égal au nombre d'états possibles d'une voie, c'est-à-dire M = 33 x (nombre de type de voies). Soit M = 33 x 2 = 66 lignes logiques.

Il suffira lorsque l'on voudra augmenter la diversité des voies traitées de rajouter pour chaque nouvelle procédure 33 lignes à cette matrice de décision.

Si le nombre des lignes est parfaitement déterminé le nombre des colonnes est beaucoup plus difficile à définir.



Tout d'abord on peut remarquer que les deux paramètres de service pour une voie sont le caractère CR_{0-7} et le bit S (ou bit d'état), le rôle du bit S est d'indiquer la nature du caractère émis CR c'est ainsi que l'on considère en fait deux demi-matrices suivant la valeur du bit S et que l'on ne garde comme paramètre d'action que le caractère (CR) ainsi (S_0 S_1 S_2 S_3 S_4, C, T) servent à déterminer la ligne et S la demi matrice sur laquelle on travaille.



Cette disposition présente un double avantage, d'une part avec cet adressage direct de la matrice et de l'état le processus d'analyse du préprocesseur n'a plus à considérer qu'une "demi-ligne" ou plutôt une partie de ligne ce qui minimise notoirement le travail du processus séquentiel d'analyse de tous les cas possibles, d'autre part il y a un gain certain sur la dimension de ces matrices.

Si l'on raisonne au niveau d'une ligne (donc d'un état), sur une demi-matrice donnée, on a dit qu'étant dans un état donné un abonné ne pouvait envisager qu'un petit nombre d'actions prévues dans le cadre du schéma de signalisation, ce qui revient à dire que une voie étant dans un état particulier le préprocesseur s'attendra à recevoir sur cette voie un caractère pris parmi un petit jeu de caractères possibles le jeu étant différent suivant que le bit S reçu soit ON ou OFF. C'est ainsi que dans chacune des demi-matrices on réservera une colonne pour chaque caractère possible.

Soient nj et n'j les nombres respectifs d'éventualités retenues pour l'état Sj dans les deux demi-matrices, dans le cas présent le nombre de colonne de la matrice S-ON doit être : $N_{-S\emptyset N} = MAX\ (Nj),\ \forall j$

et celui de la matrice S-ØFF :

$$N\text{-}S\emptyset FF = MAX\ (N'j),\ \forall j$$

En effet, il n'y a aucune raison pour que d'un état à un autre le nombre de cas possibles attendus soit le même, et même pour un état donné il diffère très certainement dans l'un et l'autre des deux demi-matrices et cependant on est contraint de dimensionner ces deux matrices compte tenu du maximum de cas possibles non seulement pour un schéma de signalisation donné mais aussi pour toute procédure.

Donc dans le cas général ces matrices de décision seront des matrices creuses, qui sont extrêmement coûteuses à représenter si on les considère comme pleines, c'est pourquoi les tables de décision du préprocesseur sont conçues physiquement pour la représentation et le traitement "économiques" des matrices creuses, ce qui permet en outre de considérer en théorie pour un état donné une infinité de cas possibles (ce dispositif est décrit dans le document concernant le matériel).

L'élément de la matrice de décision

L'élément de la matrice de décision est l'information extraite à chaque cycle par le processus d'analyse séquentielle ce mot ou case de la matrice doit comprendre nécessairement le caractère attendu (Cj), dans ce cas particulier, et une décision qui peut être un traitement local effectué entièrement par le préprocesseur ou bien l'envoi d'une demande de traitement à l'UT microprogrammée, ou bien encore les deux, la définition exacte d'un élément de la matrice de décision est donnée en annexe (III), nous en donnons ci-dessous une représentation schématique.

CAT_{0-7}	HS	CAR1	NLAV	CE_{0-7}	$MP1_{0-7}$	CAR2	$MP2_{0-7}$	C1

CAT : caractère attendu.

(HS-CAR1-NLAV CAR2) : = Sont des indications sur la nature du caractère à émettre sur la voie.

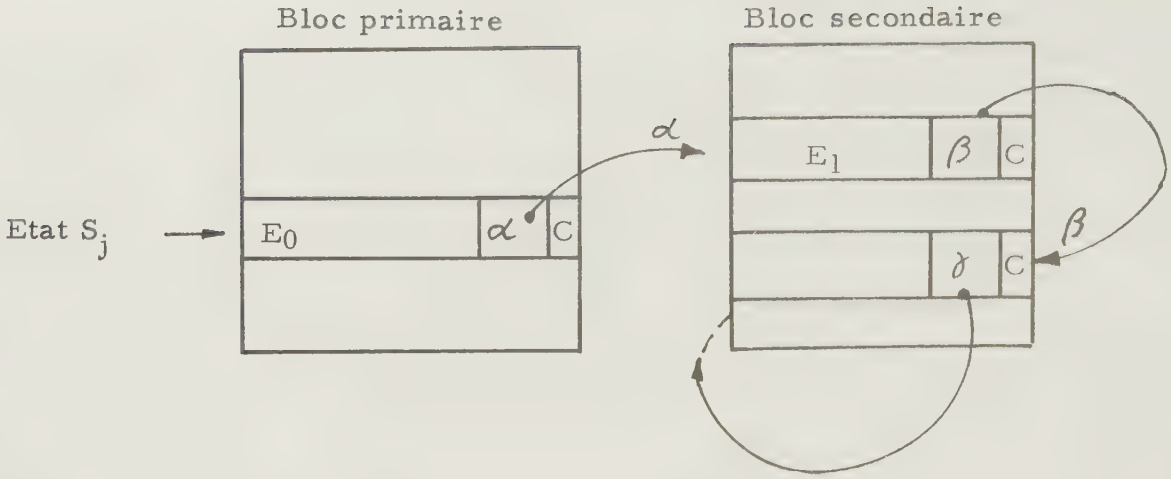
MP1 : = est le code d'une tâche à activer dans le microprocesseur dans le cas où le caractère attendu est bien le caractère reçu.

C1 : = est une indication de chainage qui indique au processus séquentiel qu'il doit poursuivre son analyse ou bien s'arrêter.

MP2 : = Dans le cas où le processus séquentiel doit s'arrêter MP2 est le code d'une tâche à activer dans le microprocesseur.

Dans le cas de chainage (indiqué par C1) MP2 est un lien de chainage.

Physiquement les éléments d'une même matrice sont situés dans deux blocs mémoires différents, le bloc primaire et le bloc secondaire (voir figure ci-dessous).



Le bloc primaire contient les éléments correspondant au premier cas à envisager par le processus séquentiel, les éléments correspondants aux autres cas sont mémorisés dans un second bloc mémoire : le bloc secondaire où ils sont chainés au moyen de l'artifice exposé plus haut. Ainsi au lieu de lire toute une ligne d'un bloc on lit tout d'abord le premier élément puis on recherche les éléments suivants au fur et à mesure que l'on en a besoin, ce "repliage" des lignes permet de représenter ainsi plus avantageusement une matrice creuse de grande dimension.

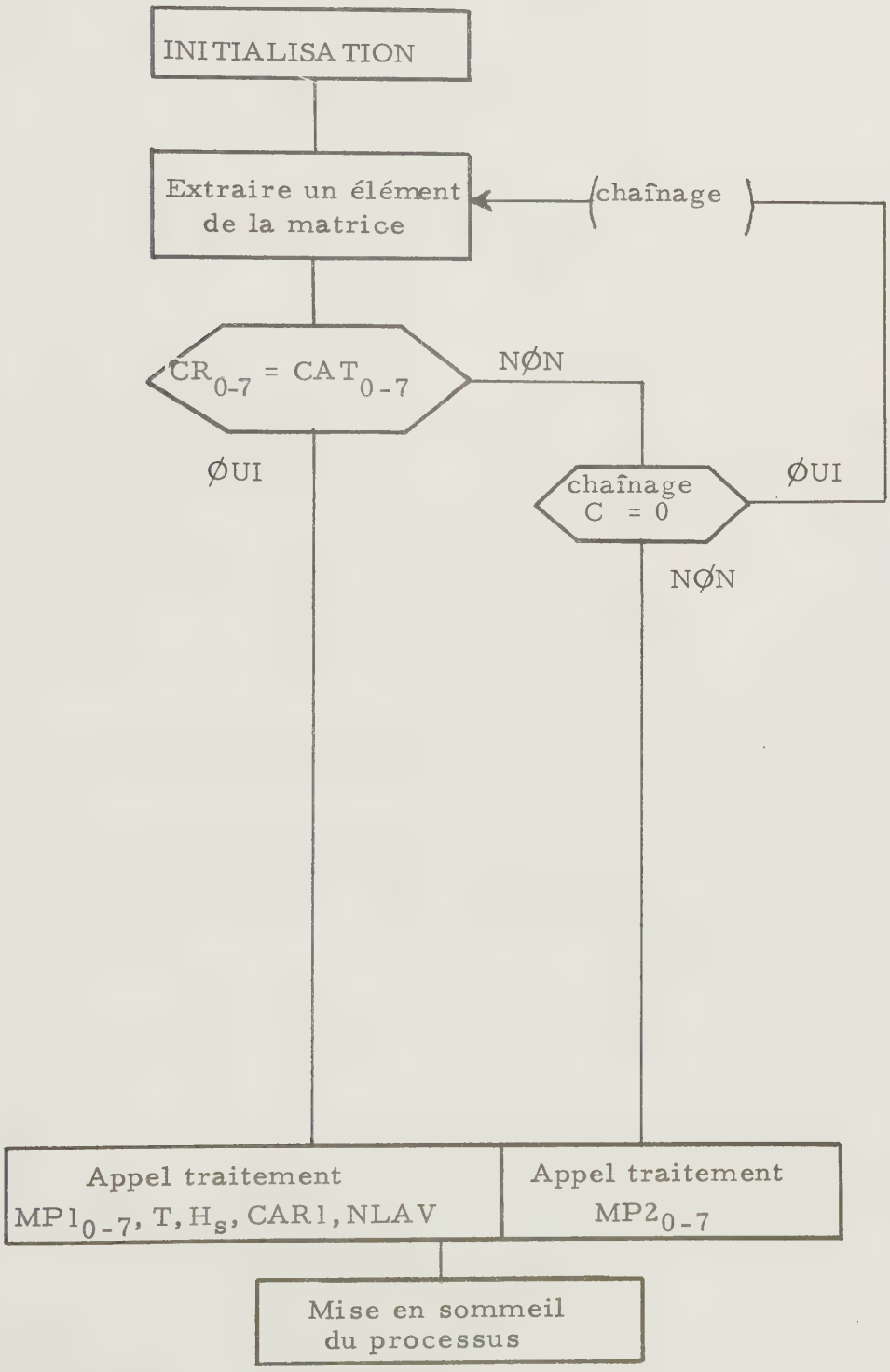
Le processus d'analyse séquentielle

On voit maintenant que le contenu de la matrice de décision (donc du bloc primaire et du bloc secondaire) constitue en quelque sorte le programme du préprocesseur, et il suffit de modifier le contenu de cette matrice pour adapter le préprocesseur à un nouveau schéma de signalisation.

Le processus d'analyse qui opère la gestion de la matrice est lui complètement câblé, on peut maintenant en donner un organigramme plus détaillé.

LE PROCESSUS D'ANALYSE SEQUENTIELLE

-:-:-:-:-:-:-:-



Ce processus d'analyse séquentielle a pour seule tâche d'extraire un élément de la matrice de décision et de comparer le caractère contenu dans cet élément (caractère attendu) avec le caractère reçu (CR). Au cas où la comparaison s'avère positive le processus passe ensuite la main au processus de traitement qui exécutera le traitement local spécifié et éventuellement enverra un message à l'unité de traitement microprogrammée afin d'y activer une tâche précise. Si la comparaison s'avère négative, le processus considère l'indication de lien de chainage, si il y a chainage le processus boucle sur lui-même et accède à l'élément désigné par le lien de chainage sinon, il s'interrompt et passe la main du processus de traitement qui enverra obligatoirement un message à l'unité de traitement microprogrammée pour l'activation d'une tâche.

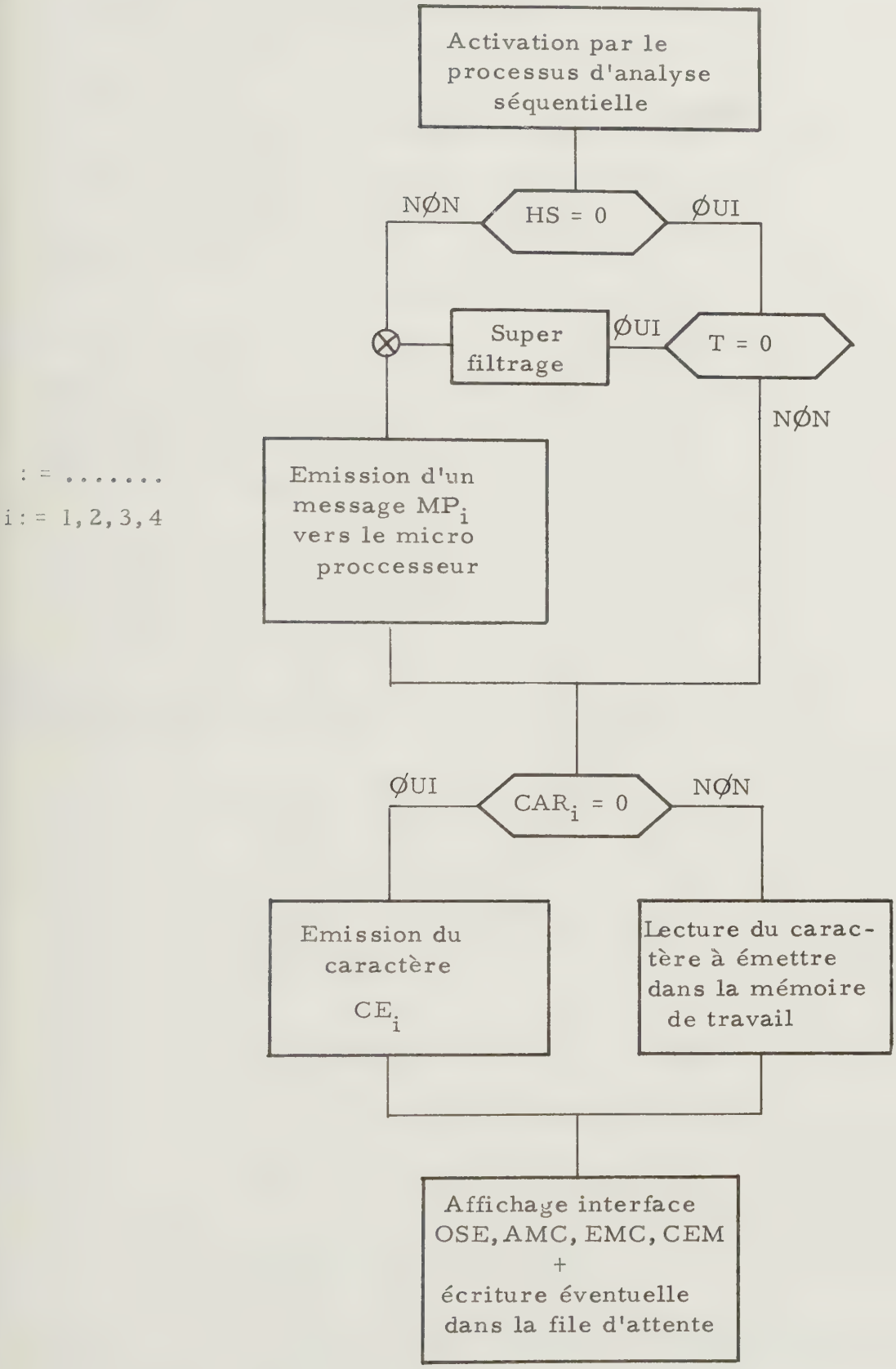
2) - La fonction traitement :

Le processus d'analyse est un processus synchrone il est initialisé par un moyen physique (une base de temps) qui lui fournit le premier élément d'une ligne d'une matrice et il explore successivement tous les éléments de cette ligne jusqu'à ce qu'il trouve celui qui correspond au cas qu'il traite, il s'interrompt alors et active le processus de traitement qui lui est de nature asynchrone, c'est-à-dire qu'il s'exécute une seule et unique fois et qu'il peut être déclenché à tout moment par le processus d'analyse.

Son rôle est de décoder les indications relatives au traitement indiqué dans l'élément de décision.

Ce processus est entièrement câblé, on ne peut donc le modifier par programme. Son organigramme est donné ci-dessous.

LE PRØCESSUS DE TRAITEMENT



Tous les sigles employés dans cet organigramme sont explicités dans l'annexe III qui décrit précisément le mot du bloc primaire (BP) du bloc secondaire (BS) et le message inscrit dans la file d'attente de communication avec l'unité de traitement microprogrammée.

Nous n'entrerons pas dans le détail de cet organigramme, disons simplement que la fonction traitement a pour rôle :

1) - De décider si le traitement est purement local (HS = 0) ou bien doit s'accompagner de l'envoi d'un message à l'UT microprogrammée (HS = 1) c'est en quelque sorte la réalisation de la fonction de filtrage.

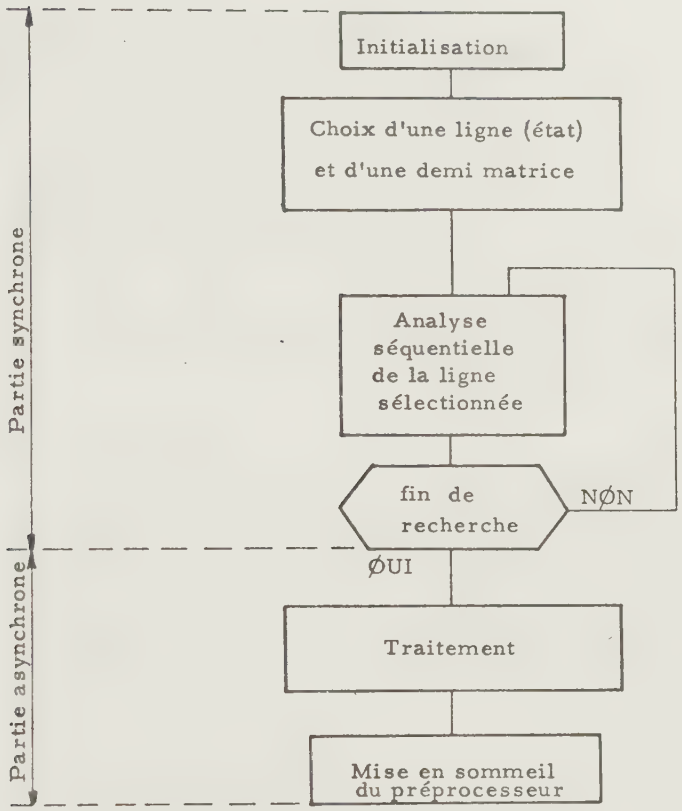
2) - De réaliser la fonction de super filtrage c'est-à-dire lorsque HS = 0 et T = 0 et que la file d'attente atteint un niveau critique de remplissage, le message n'est pas envoyé (ce cas est utilisé lors de la détection de nouveaux appels qu'on met en attente) ceci ne devant dégrader que très peu le service.

3) - De déterminer quel est le caractère à émettre sur la voie qui peut être celui fourni par le processus d'analyse séquentielle ou bien un caractère lu dans la mémoire de travail.

4) - D'assembler l'octet à émettre, d'afficher l'interface et éventuellement d'inscrire un message dans la file d'attente.

3) - La fonction filtre :

Ce qui vient d'être dit peut être résumé par l'organigramme ci-dessous :



La base de temps du réseau de connexion (BT) active au début de chaque IT un processus qui à partir de l'état de la voie et du bit S fait le choix d'une ligne et d'une demi matrice de décision. Un mécanisme de recherche séquentielle analyse ensuite systématiquement la ligne de la demi-matrice choisie qui lorsqu'il a trouvé le cas réel déclenche le processus de traitement.

Dans le schéma précédent la fonction filtre apparaît comme programmée, en effet, c'est d'après le contenu de l'élément de décision (le mot du bloc primaire ou du bloc secondaire) retenu par le processus d'analyse séquentielle, que le processus de traitement décide ou non d'émettre un message vers l'unité de traitement microprogrammée. L'efficacité de cette fonction est donc étroitement dépendante de la génération de la matrice de décision, on trouvera au dernier paragraphe de ce chapitre quelques considérations sur l'efficacité de cette programmation.

Le message inscrit dans la file d'attente de communication avec l'unité de traitement microprogrammée est décrit en détail dans l'annexe (III), il comprend sur 32 e.b. le caractère reçu

message file d'attente

0		31
Opérande	MP	CR

sur la voie, le code d'une tâche à activer dans le microprocesseur et un opérande qui est soit l'adresse de la voie concerné par ce message soit le numéro du microenregistreur associé à la voie et une indication de la phase du dernier octet émis.

Ce message est le résidu du filtrage effectué par le préprocesseur, il contient toute l'information utile à l'échelon supérieu dans la hiérarchie des organes de commande.

III. 2. - La génération des tables de décision

2.1. - Un cas concret :

Le plus simple pour faire comprendre en quoi consiste et comment se fait la programmation du préprocesseur est de prendre un exemple concret qui permette de suivre la démarche du programmeur.

Nous prendrons pour exemple le schéma de signalisation défini dans l'annexe (II), nous avons donné précédemment le diagramme d'état de cette procédure.

Le principe de la programmation du préprocesseur est de partir de ce diagramme d'état et de distinguer dans chacune des transactions d'un abonné avec l'organe de commande plusieurs étapes ou états intermédiaire

Par exemple la figure de la page suivante décrit le passage d'un abonné de l'état de ligne libre à celui de communication et l'on voit que les étapes successives sont :

a) - ligne libre

b) - attente d'invitation à numéroter

c) - invitation à numéroter

d) - attente de progression d'appel

e) - émission du caractère de progression d'appel

f) - attente de connexion

g) - communication

C'est l'UMS qui provoque les chargements d'état d'une voie en modifiant le mot de commande associé mais l'indication de changement d'état doit être contenue dans l'élément de décision extrait de la matrice. C'est à dire que partant d'un état donné, la décision prise si elle consiste en l'envoi d'un message à l'unité de traitement microprogrammée, doit être de donner à celle-ci suffisamment d'information pour lui permettre de faire passer la voie dans l'état suivant.

Etats	S	CAT 0	Decision	CAT 1	Decision	CAT 2	Decision	CAT 3	Decision	CAT 4	Decision	CAT 5	Decision
Ligne libre	off	$\hat{0}$	Local CE = $\hat{0}$	$\neq \hat{0}$	Soft erreur								
	on	1/6	Soft CE = $\hat{0}$	$\neq$ 1/6	Soft erreur								
attente d'invitation à numeroter	off	tout caractère	Soft : Liberation										
	on	1/6	Local CE = $\hat{0}$	$\neq$ 1/6	Soft erreur								
Invitation à numeroter	off	tout caractère	Soft Liberation										
	on	1/6	Local CE = 2/11	2/11	Soft. fin de selection	$\neq$ 1/6 $\neq$ 2/11	Soft. caractère selection						
attente progression d'appel	off	tout caractère	Soft Liberation										
	on	1/6	Local CE = 1/6	$\neq$ 1/6	Soft erreur								
Emission du caractère de progression d'appel	off	tout caractère	Soft Liberation										
	on	1/6	Soft. CE = ---	$\neq$ 1/6	Soft erreur								
Attente connexion	off	tout caractère	Soft. Liberation										
	on	1/6	Local CE = 1/6	$\neq$ 1/6	Soft : erreur								
Communication	off	0/3	Soft Rappel immédiat	$\neq$ 0/3	Soft Liberation								
	on	tout caractère	Local										
	off												
	on												
	off												
	on												

Nous allons suivre successivement chacune des étapes et voir comment on peut les transcrire en termes de la matrice de décision.

a) Ligne libre :

C'est un état dit permanent c'est à dire que l'abonné peut y rester indéfiniment et qu'aucune temporisation n'en limite la durée. Si l'on regarde le diagramme d'état de la signalisation on voit que l'alarme peut dans cet état entreprendre deux actions licites :

- rester dans l'état de ligne libre
- procéder à une demande d'appel

Dans le premier cas il doit émettre en permanence

CR = $\hat{0}$ ("tout zéro")

S = off (S = 0)

alors un traitement local suffit qui consiste à renvoyer sur la voie le caractère $\hat{0}$.
La combinaison S = off
CR $\neq \hat{0}$

correspond à une erreur et devra faire l'objet d'un traitement spécial. Tout ceci peut se coder dans la matrice de décision de la façon suivante :

Mot du bloc primaire correspondant à la demi matrice S = OFF et à l'état ligne libre.

Ce mot constitue la ligne complète de la matrice de décision associée à l'état ligne libre et au cas S-OFF.

En effet si le caractère reçu CR est bien $\hat{0}$ alors on a CR = CAT

Alors on exécute les directives qui suivent :

HS = 0 c'est à dire que le traitement est local et qu'il n'y a pas d'envoi de message du microprocesseur.

CAR_1 = 0 Le carctère émis est le caractère codé dans CE donc ici le caractère $\hat{0}$.

Si CR $\neq$ CAT

alors on exécute les directives codées dans la deuxième partie du mot :

C1 = 1 ; il n'y a pas chainage il y a obligatoirement envoi d'un message à l'UT microprogrammée ce message comprenant le code de la tâche exécutée (α) et l'adresse de voie de l'abonné (NLAV = 0).

Cet état correspond au cas le plus simple où il n'y a qu'une alternative entre deux possibilités si plusieurs actions étaient possibles à partir d'un même état la ligne de la matrice de décision aurait été plus longue et on l'aurait codée en utilisant la possibilité de chaînage dans le bloc secondaire.

Le cas de demande d'appel,

CR = 1/6 (caractère de l'alphabet CCITT n° 5).

S = ON

nécessite obligatoirement l'envoi d'un message à l'UT microprogrammée qui devra réserver un microenregistreur pour cette transaction et faire passer la voie dans l'état d'attente d'invitation à numéroter.

De même que précédemment la combinaison

CR = 1/6

S = ON

correspond à une erreur et un message sera passé à l'UT microprogrammée. L'élément de décision associé à ce cas peut être codé de la façon suivante :

Mot du bloc primaire correspondant à la demi matrice.

0	1/6	0	0	0	$\hat{0}$	β	0	α	1
T	CAT	HS	CAR 1	NLAV	CE	MP1	CAR 2	MP2	C1

qui peut s'interpréter ainsi, le caractère attendu CAT est le caractère 1/6, si le caractère reçu est bien le caractère attendu alors un message doit être écrit dans la file d'attente, le dispositif de super filtrage doit être armé (T = 0 et HS = 0) et le caractère à émettre est le caractère CE = $\hat{0}$.

Le message contiendra le code MP1 = β d'une tâche du microprocesseur et l'adresse de la voie de l'abonné (NLAV = 0).

Si le caractère attendu n'est pas le caractère reçu (CAT≠CR) alors on exécute la deuxième partie de l'élément de décision car il n'y a pas chainage (C_1 = 1), donc un message devra être inscrit dans la file d'attente et celui contiendra l'adresse de voie de l'abonné (NLAV = 0) et le code $MP_2 = \alpha$ de la tâche de traitement d'erreur.

L'envoi du message $MP_1 = \beta$ activera une tâche du microprocesseur qui fera passer la voie de l'état de ligne libre à celui d'attente d'invitationà numéroter.

b) - Attente d'invitation à numéroter :

Cette étape peut être qualifiée d'état semi permanent puisqu'une temporisation est associée à l'envoi de l'invitation à numéroter qui interdit que la voie y demeure indéfiniment. A ce stade l'abonné peut exercer deux actions :

- se libéreren faisant passer son bit S à l'état OFF.
- attendre et envoyer de façon permanente

  S = ON

  CR = 1/6

Dans le premier cas le préprocesseur doit décider l'envoi d'un message de libération à l'UT microprogrammée qui provoquera le retour de la voie à l'état de ligne libre.

Le second cas ne nécessite qu'un traitement purement local c'est à dire l'envoi d'un caractère CE = $\hat{0}$.

c) Invitation à numéroter :

C'est encore un état semi permanent, la voie a été placée dans cet état par l'UMS sur ordre de l'UT consécutivement à une demande d'appel. Un abonné dans cet état peut selon la procédure soit demander sa libération, soit envoyer ses caractères de sélection ou le caractère (2/11) de fin de sélecti ou bien encore attendre l'accusé de réception de son message de sélection, on doit donc encore ici distinguer deux cas :

- S = OFF : l'abonné demande à se libérer et un message est envoyé à l'UT microprogrammée.
- S = ON : on doit alors distinguer selon le caractère re Si CR = 1/6 alors le traitement peut être loca et consister en l'envoi du caractère CE = 2/11 (l'abonné n'a pas encore commencé sa sélectio

  Si CR = 2/11, l'abonné vient de terminer sa

sélection, un avis de fin de sélection accompagné du numéro du microenregistreur (contenu dans le mot de commande) est inscrit dans la file d'attente. Si le caractère reçu CR est différent de 1/6 et de 2/11 il est susceptible d'être un caractère de sélection, dans ce cas également un message contenant le caractère de sélection et le numéro du microenregistreur associé à la voie sera inscrit dans la file d'attente.

Le codage de chacun de ces cas ne présente aucune difficulté la seule différence avec les exemples précédents est que cette fois ci la ligne de la demi matrice SON comportera trois éléments et qu'il faudra avoir recours au chainage dans le bloc secondaire.

d) - Attente de progression d'appel :

Dès qu'elle a reçu l'avis de fin de sélection l'unité de traitement microprogrammée place la voie considérée dans l'état d'attente de progression d'appel qui est également un état semi permanent qui lui permet effectivement d'attendre que l'unité de traitement accomplisse toutes les tâches préparatoires à la connexion et élabore le caractère de progression d'appel.

Dans cet état l'abonné peut exercer deux actions :

- Attendre, en émettant

  S = ON

  CR = 1/6

ce qui ne nécessite qu'un traitement local avec l'émission au caractère CE = 1/6

- Demander sa libération avec

  S = OFF

Dans ce cas la demande de libération est inscrite dans la file d'attente. Ce cas peut être codé sous forme d'un seul mot du bloc primaire.

e) - Emission du caractère de progression d'appel :

C'est un état transitoire puisque le caractère de progression d'appel n'est émis qu'une seule fois sur une voie donnée, c'est à dire que le préprocesseur doit pour cet état, chercher dans la mémoire de travail de l'UMS le caractère de progression d'appel (dans le microenregistreur associé à la voie), l'émettre et donner à l'unité de traitement microprogrammée un avis d'envoi du caractère de progression d'appel.

Le préprocesseur sait qu'il doit chercher un caractère dans la mémoire de travail lorsque l'e.b. CAR1 de l'élément de décision sélectionné est à 1 alors ce caractère est à chercher dans la mémoire de travail dans le microenregistreur associé à la voie dont le numéro est codé dans le mot de commande (LMC).

f) - Attente de connexion :

Lorsque le caractère de progression d'appel est émis l'UT microprogramme place la voie dans l'état d'attente de connexion. Le codage de l'élément de décision associé à cet état ne diffère en rien de l'état d'attente de progression d'appel dont on a parlé précédemment et on pourra même utiliser le même emplacement physique de mémoire dans le bloc primaire pour coder les deux états.

g) - Communication

L'état de communication est un état permanent qui est l'aboutissement de la procédure d'établissement d'un circuit. Deux cas peuvent se présenter à partir de cet état :

- S = ON, alors le préprocesseur n'a pas à tenir compte du caractère reçu -inhibition du test sur le caractère reçu), l'abonné demeure en communication, aucun traitement n'est nécessaire.
- S = OFF, signifie que l'abonné vient demander un service au commutateur, il y a alors lieu de regarder le caractère reçu. Si CR = O/3 l'abonné demande la libération de la communication en cours mais annonce un rappel immédiat, ceci se traduit par l'envoi d'un message comportant l'adresse de la voie et la notification du rappel immédiat. Si CR $\neq$ 0/3 le cas est interprété comme une demande de libération et un message est inscrit dans la file d'attente.

Ce qui précède n'est qu'un exemple très fragmentaire de programmation des tables de décision qui montre néanmoins que la structure choisie permet une transcription presque immédiate du diagramme d'état du type de signalisation choisi.

2.2. - Le programme de génération des tables de décision :

Les tables de décision sont réparties physiquement dans trois blocs mémoires distincts appartenant au préprocesseur, ce sont le bloc transcodeur (BT) qui à partir des paramètres d'état (S_0, S_1, S_2, S_3, S_4, C, T, S) sélectionne la demi-matrice de décision et la ligne utile, le bloc primaire et le bloc secondaire dont on a déjà parlé.

Lors de l'initialisation du système, le contenu de ces tables (image mémoire) est disponible dans la mémoire vive de l'unité de traitement sous forme d'un fichier que le programme d'initialisation de l'unité de traitement microprogrammée lit séquentiellement puis l'envoie mot par mot au préprocesseur avec une commande d'inscription.

Le départ de la chaine d'initialisation étant l'unité de traitement il était logique que la génération des tables de décision soit faite au niveau de l'UT.

C'est pourquoi nous avons défini un langage (GENTADE) permettant une représentation symbolique de la matrice de décision, le programme de gestion de ce langage est supporté par l'unité de traitement.

Le programme fonctionne selon le mode conversationnel, il permet à l'utilisateur de spécifier la demi-matrice de la ligne qu'il veut définir puis de décrire cette ligne sous forme symbolique, la génération des mots du bloc transcodeur, du bloc primaire et du bloc secondaire étant faite automatiquement par le programme. Les commandes de ce programme sont décrites dans l'annexe IV.

Le programme GENTADE est un outil de mise au point utile même nécessaire mais il faut toutefois remarquer qu'il n'est justement qu'un outil de transcription et qu'il n'offre aucune possibilité d'optimisation du code généré, c'est à dire que l'utilisation la plus rationnelle des tables de décision ne peut résulter que de l'habileté du seul programmeur (du préprocesseur).

2.3. - Quelques considérations sur l'utilisation rationnelle des tables de décision :

On a dit au début de ce chapitre que l'on avait voulu situer le préprocesseur, considéré comme processus, entre deux cas limites. La limite inférieure correspondant au filtre minimum est imposée par la puissance de l'unité de traitement microprogrammée la limite supérieure nous a été imposée par des considérationssur les coûts en temps et en matériel nécessaires à la réalisation d'un système complexe. En définitive ayant choisi un moyen terme nous l'avons partiellement figé dans une structure matérielle qui conserve cependant la notion de programmation.

Le préprocesseur est certes un outil rudimentaire comparé aux possibilités de l'unité de traitement microprogrammée mais il possède l'avantage de pouvoir effectuer aisément et rapidement des comparaisons de caractères qui sont extrêment coûteuses en temps pour un processeur programmé.

Le souci constant du programmeur des tables de décision sera donc de décharger autant que possible l'unité de traitement microprogrammée de toute opération de comparaison de chaine de caractères et de la recherche d'un caractère dans une chaine. Ceci suppose que l'unité de traitement

microprogrammée n'ait à procéder à aucune analyse de l'état d'une voie ce qui signifie que le préprocesseur doit lui envoyer dans chaque message le numéro de la tâche élémentaire précise à exécuter.

Ceci peut être réaliser au niveau du préprocesseur par deux moyens :

- d'une part l'utilisation maximum de l'éventail des états possibles donc la découpe du diagramme d'état d'une procédure en un nombre maximal de sous états auquels on peut associer des tâches élémentaires spécialisées.
- d'autre part pour un état donné il est possible de prévoir un nombre variable d'éventualités (de caractères attendus) plus ce nombre est grand et plus fine est l'analyse de la demande de l'abonné.

C'est donc en jouant sur ces deux paramètres que le programmeur pourra optimiser ou plutôt rationaliser l'emploi de sa table de décision dans la limite des possibilités matérielles soit 33 états possibles pour une voie et pour une procédure particulière et 128 mots disponibles en tout et pour tout dans le bloc secondaire.

-:-:-:-:-:-:-:-:-:-

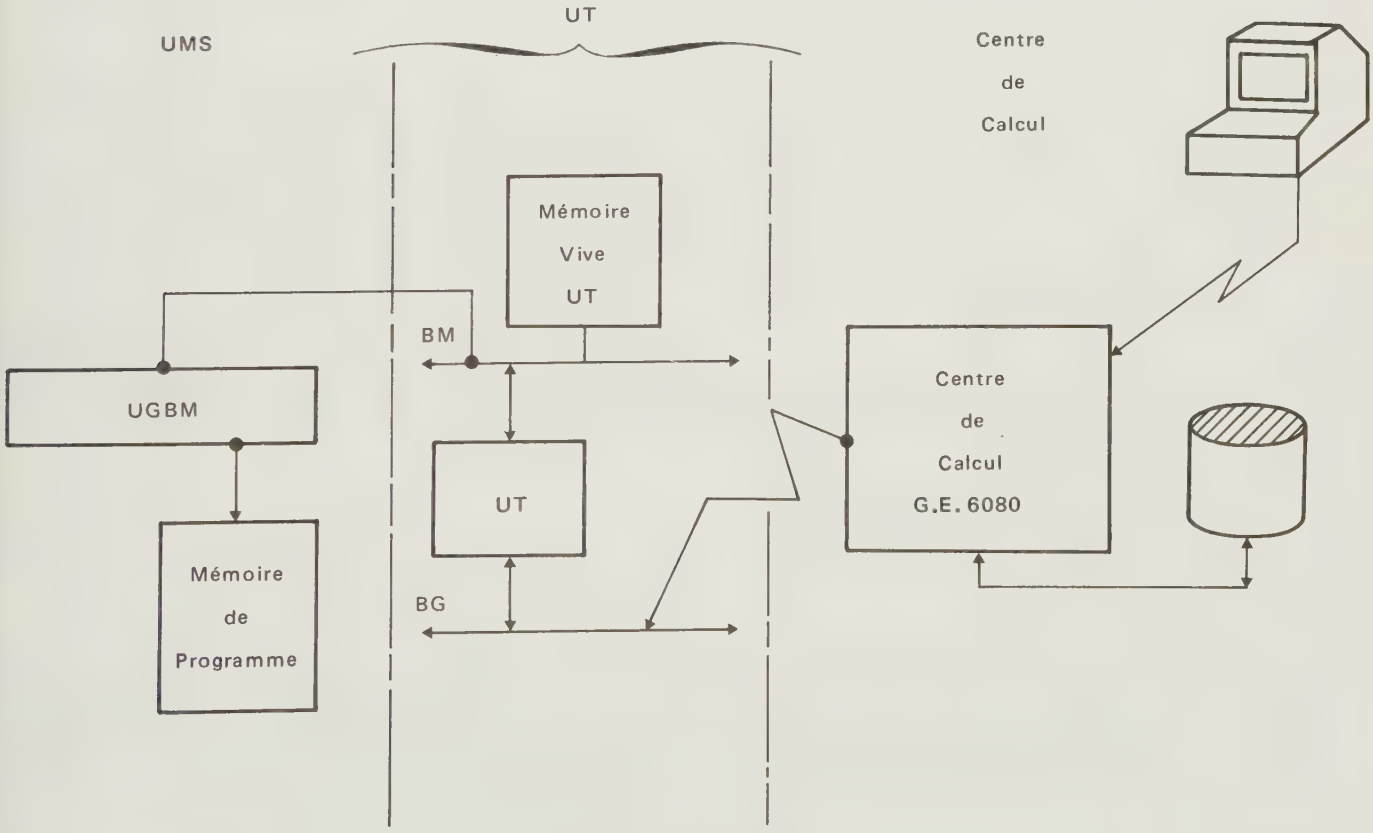
IV - LA PROGRAMMATION

IV.1 - Les outils de programmation

L'UMS est un organe microprogrammé; son programme est inscrit dans une mémoire spécialisée (mémoire de programme - MP) dont la structure et l'utilisation sont décrites dans le document concernant le matériel (chapitre IV).

Chacune des microinstructions est codée dans un mot mémoire de 32 éléments binaires et la taille maximale du programme opérationnel est de 4096 (4 K) mots. Ces deux contraintes rendent nécessaire la mise en oeuvre d'outils de programmation permettant la génération du fichier binaire objet correspondant au microprogramme ; il était en effet, inconcevable que le microprogrammeur ait à générer e.b. par e.b., et à inscrire en mémoire par un moyen quelconque ce fichier binaire.

La procédure retenue pour la génération des microprogrammes peut être qualifiée d'indirecte puisqu'on utilise le support logiciel du centre de calcul du CNET (un ordinateur G.E. 6080) et de l'unité de traitement de la maquette de commutation (un calculateur T 1600).



Le premier outil de programmation développé est un microassembleur. C'est un programme disponible sur l'ordinateur au centre de calcul du CNET qui permet au microprogrammeur d'écrire son microprogramme sous une forme symbolique ; il peut ainsi utiliser un langage de programmation qui lui ôte le souci de raisonner au niveau de l'élément binaire.

Ce programme d'assemblage se compose, en fait, d'un assembleur (MICRASS) qui génère des fichiers binaires translatables (un pour chacun des modules du programme final) et d'un éditeur de liens qui réunit et chaîne les différents modules (binaires translatables) en un fichier binaire absolu (le binaire objet) qui est l'image exacte du programme qui sera implanté dans la mémoire de l'UMS.

Ce fichier binaire absolu (BØ) est stocké dans une mémoire permanente de l'ordinateur du centre de calcul ; il peut y être lu et envoyé (8 e.b. par 8 e.b.) vers n'importe quel périphérique et en particulier, vers l'unité de traitement de la maquette de commutation, car l'UT est connectée directement au centre de calcul (voir figure ci-dessus).

Un programme de l'UT - le chargeur - acquiert le fichier binaire objet (8 e.b. par 8 e.b.) et l'inscrit séquentiellement dans la mémoire vive sous forme de mots de 16 e.b.

En définitive, le fichier séquentiel en mémoire vive est lu par un microprogramme (Bootstrap) résidant dans la partie mémoire morte (RØM) de la mémoire de programme de l'UMS; les mots de 16 e.b. sont assemblés deux par deux, sous forme de mots de 32 e.b. et chargés séquentiellement dans la mémoire de programme du processeur microprogrammé :

Ainsi, l'ensemble des outils de programmation nécessaire à la génération et au changement des microprogrammes se limite à trois composantes :

- le microassembleur,
- l'éditeur de liens,
- le chargeur.

Nous parlerons plus particulièrement du microassembleur et de l'éditeur de liens comme étant les plus originaux mais,auparavant, il paraît utile de définir le langage d'assemblage utilisé par le microprogrammeur.

1.1 - Le langage d'assemblage

La structure du mot de la microinstruction est résumée dans le tableau 1; sa description détaillée est donnée en annexe (V). Chaque microinstruction peut être

Code d'Ordre 2 bits	Opération interne UT — Opération principale				Opérations secondaire			Commandes Externes	Adresse de μ instruction suivante
CALCUL	(Registre 1ère opérande)	Opération Add,A,V,Q	(Registre 2ème opérande)	→ Registre Destinataire	Report initial 0,1 Rp	Posit^t de Rp et C / Posit^t de T	4 bits Trait^t N° de bit Emploi de RMOD	4 bits Commandes UGBM	2 bits – Séquence – Emploi de TAP – Rupture Conditionnelle
	5 bits	2 bits	2 bits	5 bits	2 bits	4 bits	Adresse de rupture 8 bits		
DÉCALAGES	(idem)	Opération Droite 1 pas – 8 – Gauche 1 – – 8 –	(idem)	→ (idem)	(idem) 0,1,Rp,L10	(idem)	(idem) Adresse de rupture	Commandes UGBM	idem
VALEUR	(Registre 1ère opérande)	Opération Add,A,V,Q	(Valeur)	→ { m̂ Registre ou RTR ou rien		(idem)	(idem) Adresse de rupture	Commandes UGBM	idem
E/S — S, 1 bit	(Registre 1ère opérande) → bus	Opération Ad,A,V,Q	(RTR ou zéro)	→ m̂ Registre	(idem) 0,1, Rp	(idem)	4 bits Traitement de N° de bit ou Emploi de RMOD	COMMANDES BUS GÉNÉRAL – Adresse – Fonction 8 bits	– Séquence – Emploi de TAP – Emploi de TAP Conditionnel
	5 bits	2 bits	1 bit		2 bits	4 bits			
E	(Registre 1ère opérande ou zéro)	Opération	Info bus	→ m̂ Registre	(idem) 0,1, Rp	(idem)			2 bits

scindée en différents champs ; leur configuration est variable suivant le type de la microinstruction, néanmoins, on trouve généralement :

- un champ indiquant le type de la microinstruction : calcul, décalage, entrée-sortie, valeur,
- un champ donnant le registre utilisé comme premier opérande
- un champ indiquant l'opération réalisée : addition, union, intersection,
- un champ indiquant le registre destinataire,
- un champ précisant le mode d'enchaînement choisi pour les microinstructions.

Certains champs, tels ceux mentionnés plus haut, sont obligatoires ; d'autres, facultatifs, seront pris par défaut. Il est donc nécessaire, dans la définition de langage symbolique, d'en tenir compte, le langage recherché devant être très dépendant de la structure de la machine.

On a donc été conduit à définir un langage qui puisse associer à chacun des champs d'une microinstruction une représentation symbolique quelconque.

La solution retenue consiste à définir un vocabulaire de mots-clés, ceux-ci étant alors non utilisables pour les symboles des différents microprogrammes. Une instruction symbolique est alors composée d'une suite de mots-clés et de symboles, séparés par des virgules, et terminés par un point-virgule. Une zone de commentaire peut être insérée après le point-virgule de fin d'une instruction symbolique. La structure type de celle-ci est alors :

|étiquette| cho^1, cho^2, ..., cho^n, chf^1, ..., chf^p ;
<commentaire>

où :

- <u>étiquette</u> : représente un symbole étiquette facultatif,
- <u>cho^i</u> : représente un champ obligatoire dont le contenu est soit un mot-clé, soit un symbole défini dans un microprogramme, soit une valeur décimale ou binaire, suivant les cas,
- <u>$ch\ f\ j$</u> : représente un champ facultatif qui, s'il est absent, sera pris par défaut égal à un certain mot-clé,
- <u>commentaire</u> : représente la zone de commentaires, facultatifs, obligatoirement précédée de < et suivie de> .

Outre le symbolique des microinstructions, il était nécessaire de se donner des pseudo-instructions permettant de définir les symboles utilisés, d'indiquer les début et fin de programmes, et enfin, de se donner la possibilité de définir des macroinstructions. En effet, les différents programmes utilisés dans la phase opérationnelle devront réaliser de manière répétitive un certain nombre d'opérations. Il était donc utile, du point de vue de la programmation, d'avoir la possibilité d'utiliser des macroinstructions, l'assembleur se chargeant lui-même de remplacer la pseudo-instruction faisant appel à une macroinstruction par la séquence de microinstructions correspondant à celle-ci.

Un microprogramme type peut alors avoir la structure suivante :

a) . Début du programme,

b) . Définition des symboles utilisés dans le programme,

c) . Définition de macroinstructions,

d) . Instruction,

e) . Fin de programme.

On trouvera en |19| la description détaillée de la syntaxe du langage d'assemblage.

1.2 - Le micro assembleur

Le programme supportant le langage d'assemblage décrit au chapitre précédent est le microassembleur.

Son rôle est de traduire le texte du programme écrit en langage symbolique en un fichier binaire translatable.

Le microassembleur assume trois types de tâches différents :

- l'analyse syntaxique du texte du programme en langage symbolique,
- la création d'une table des symboles,
- la génération du code (binaire translatable).

L'assembleur réalisé est un "assembleur à deux passages". Lors du premier passage, chaque symbole rencontré, qui ne fait pas partie du vocabulaire des mots réservés, est inscrit dans la table des symboles.

Au second passage, ce symbole sera remplacé par sa valeur, celle-ci ayant été inscrite au préalable dans la table des symboles.

On trouvera en annexe (VI) une liste de l'assembleur.

. L'analyseur syntaxique :

Le module d'analyse syntaxique constitue en fait le corps de l'assembleur, les autres modules se greffent sur celui-ci. Il comporte quatre parties principales :

- définition des éléments, des mots-clés,

- séparation des instructions, constitution des différents champs,

- analyse de chacun des champs isolés, détection des erreurs de syntaxe et génération d'une liste d'erreurs,

- enchaînement des différents modules nécessaires à la suite du traitement. L'utilisateur du microassembleur a en effet le choix entre plusieurs options : assemblage conversationnel, traitement par lots, sortie ou non d'une liste d'assemblage, désignation du nom et du support physique du fichier binaire translutable ... il existe donc un langage de commande qui lui permet de faire connaître au programme d'assemblage les différentes options désirées et c'est l'analyseur syntaxique qui enchaîne les modules correspondant à ces options.

. La table des symboles :

C'est un dictionnaire dans lequel le microassembleur classe tous les symboles rencontrés au cours de l'analyse syntaxique et qui ne fait pas partie de l'ensemble des mots réservés. Pour chaque symbole, cette table comprend :

- le nom du symbole,

- le nom du module de programme dans lequel il figure,

- des indications sur le type du symbole (interne ou externe) et sa position dans l'instruction (étiquette ou champ),

- la valeur relative du symbole, inscrite dès que le programme est en mesure de la connaître (cas d'une pseudo-instruction affectant une valeur à ce symbole). Sinon, cette valeur sera générée lors de l'édition de liens.

1.3 - L'éditeur de liens

A la fin de l'assemblage,d'un module, on dispose dans le fichier BINREL d'une incarnation dite bi-

naire translatable du programme à générer. C'est un fichier qui est proche de l'image du programme qui sera chargé dans la mémoire du microprocesseur à la différence que tous les symboles ne sont pas définis. En effet, les symboles à caractères relatifs, symboles externes définis dans d'autres modules ou symboles représentant des adresses, ne pourront être définis que lorsque l'on connaîtra l'adresse d'implantation du module, et la valeur des symboles externes référencés dans le programme. A BINREL, on associe l'image de la table des symboles (TABSYM) obtenue lors de l'assemblage.

D'une façon générale, le fichier que l'on chargera dans la mémoire de l'unité de traitement microprogrammée, sera composé de plusieurs modules assemblés séparément.

La génération de ce fichier définitif ou binaire absolu (dans lequel tous les symboles sont définis) est réalisée par l'éditeur de liens. C'est un programme auquel on indique la page de mémoire dans laquelle on désire implanter le programme définitif, ainsi que l'ordre et le nom des différents modules à chaîner dans cette page.

L'éditeur de liens parcourt les tables des symboles des différents modules à chaîner et, pour chaque symbole rencontré, il transforme la valeur relative en valeur absolue.

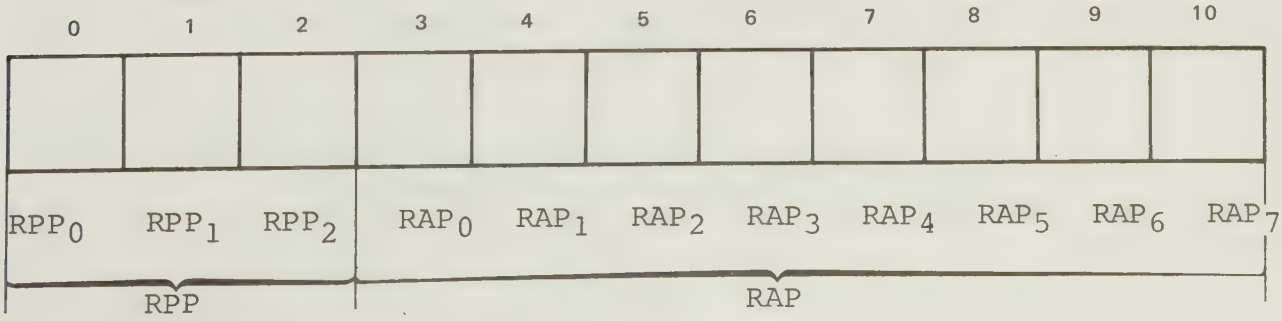
A la fin de l'édition de liens, on dispose d'un fichier représentant l'image exacte du fichier qui sera implanté dans la mémoire du microprocesseur.

IV.2 - Analyse fonctionnelle du logiciel

2.1 - La pagination

On a montré que le volume de programme nécessaire à la réalisation de la fonction UMS (le volume maximum a été évalué à 4 K mots) dépassait les limites de l'espace adressable par le microprocesseur.

En effet, le compteur ordinal (RAP) de la micro-machine est un registre de 11 e.b. qui a la structure suivante :



Il est divisé en deux champs: un champ RPP de 3 e.b. qui désigne une page parmi 8 et un champ RAP de 8 e.b. qui désigne une adresse (une parmi 256) dans la page RPP.

L'espace adressé par ce compteur ordinal est donc de :

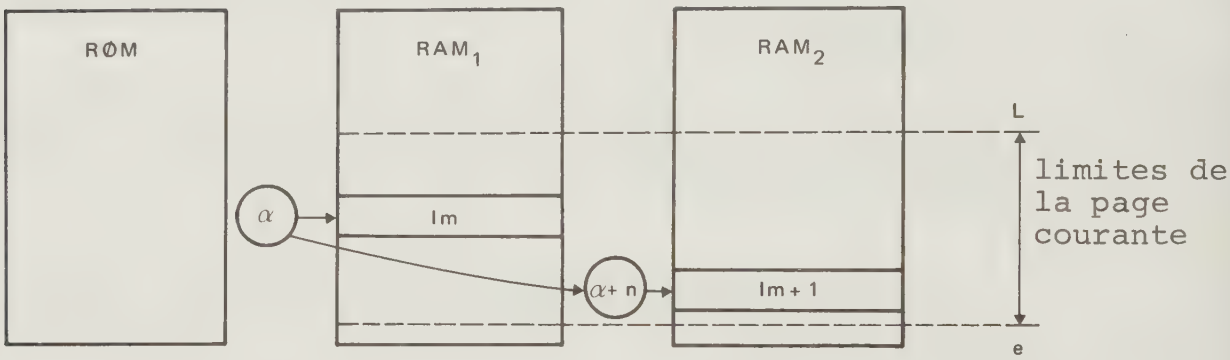
- $2^{11} = 8 \times 256 = 2048$ mots, soit 2 K mots,

ce qui est insuffisant pour notre application.

C'est pourquoi la mémoire de programme est divisée en trois blocs, chacun de ces blocs ayant une capacité théorique de 2 K mots. Ces trois blocs sont :

- le bloc RØM : en mémoire morte donc non volatile, qui sera réservé aux programmes d'initialisation,

- les blocs RAM_1 et RAM_2 de mémoires volatiles, donc réinscriptibles à volonté, qui seront utilisés pour les programmes opérationnels.



Cette structure étant définie, se pose au microprogrammeur le problème du saut de bloc. En effet, dans un bloc donné, le programmeur a la possibilité d'effectuer des branchements ou des sauts dans la limite de la page de l'instruction courante, et il peut, lors d'une instruction sur valeur, demander un changement du bloc courant et accéder ainsi à n'importe quel bloc.

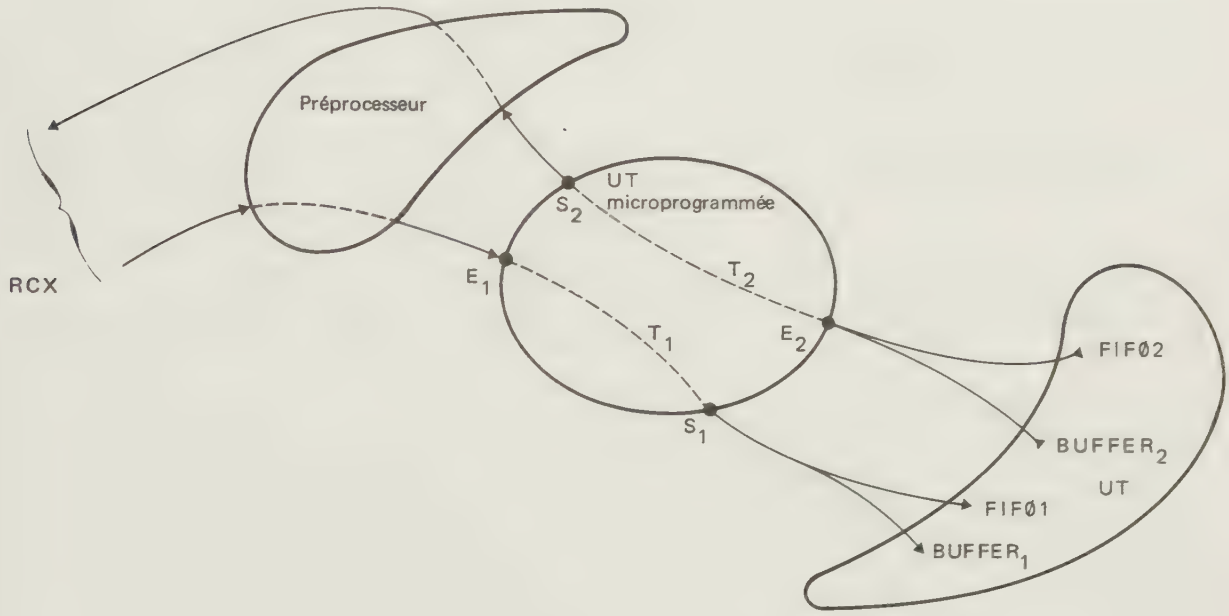
Cela signifie que si α est l'adresse de l'instruction courante, l'instruction suivante pourra être celle d'adresse $\alpha + n$ (voir figure ci-dessus) d'un autre bloc avec la condition minimale : $n \leqslant 255$ et que $(\alpha + n)$ appartient à la même page que (α).

Cette remarque d'ordre pratique sera néanmoins contraignante en ce sens que la structure même du logiciel de l'UMS en sera influencée.

2.2 - Structure du programme de l'UT microprogrammée

- le programme moniteur

L'organigramme du programme moniteur a pratiquement été défini au chapitre II concernant les relations fonctionnelles. On a vu que la fonction principale de l'unité de traitement microprogrammée était la gestion des points d'entrée E_1 et E_2 et des points de sortie S_1 et S_2 (voir figure ci-dessous).



Le programme moniteur est celui qui doit assurer l'enchaînement des différentes tâches concourant à la réalisation de la fonction unité d'échange.

Reprenons le diagramme d'état du processus de gestion des échanges déjà donné au chapitre II. (figure page suivante).

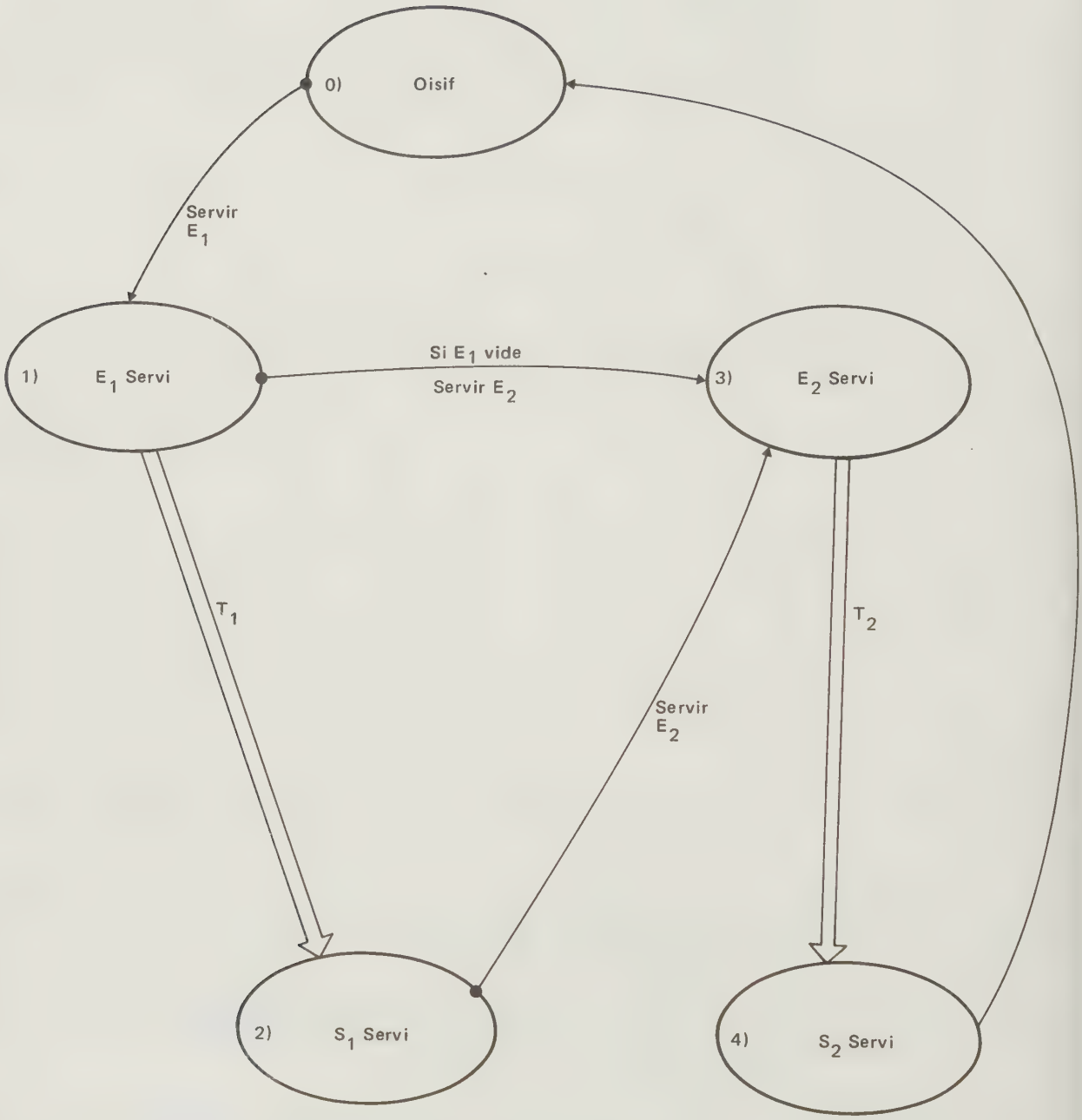
On a vu que la partie opérationnelle du programme du microprocesseur pouvait être scindée en deux tâches : T_1 et T_2 :

. T_1 a pour rôle de lire un mot dans la file d'attente du prépocesseur et d'exécuter l'ordre codé dans ce mot. L'action entreprise pourra avoir plusieurs conséquences :

. Aucun échange avec l'UT
c'est par exemple, le cas lorsqu'un caractère de sélection est reçu sur une voie et stocké dans la mémoire de travail,

. Envoi d'un message à l'UT, soit inscription d'un

Diagramme d'état du processus de gestion des échanges



mot dans FIFØ1,

- Ecriture d'un mot dans une zone mémoire ($BUFFER_1$) de l'UT (caractères de sélection ou message d'un abonné demandant une transmission différée),

- Lecture d'un mot dans une zone mémoire ($BUFFER_2$) de l'UT, c'est par exemple le cas lors de la diffusion différée d'un message.

T_1 peut donc conduire, soit à une sortie sur S_1, ou bien à une entrée sur E_2 ($BUFFER_2$).

- T_2 lit un mot dans la file d'échange UT vers UMS et exécute l'ordre qui y est codé. Il peut avoir deux conséquences :

  - Aucune sortie d'informations si le mot lu est un message de gestion interne (attribution d'un super-enregistreur),

  - Sortie d'un ou plusieurs mots sur S_2 lorsque l'ordre reçu consiste en la modification d'un mot de la mémoire de commande.

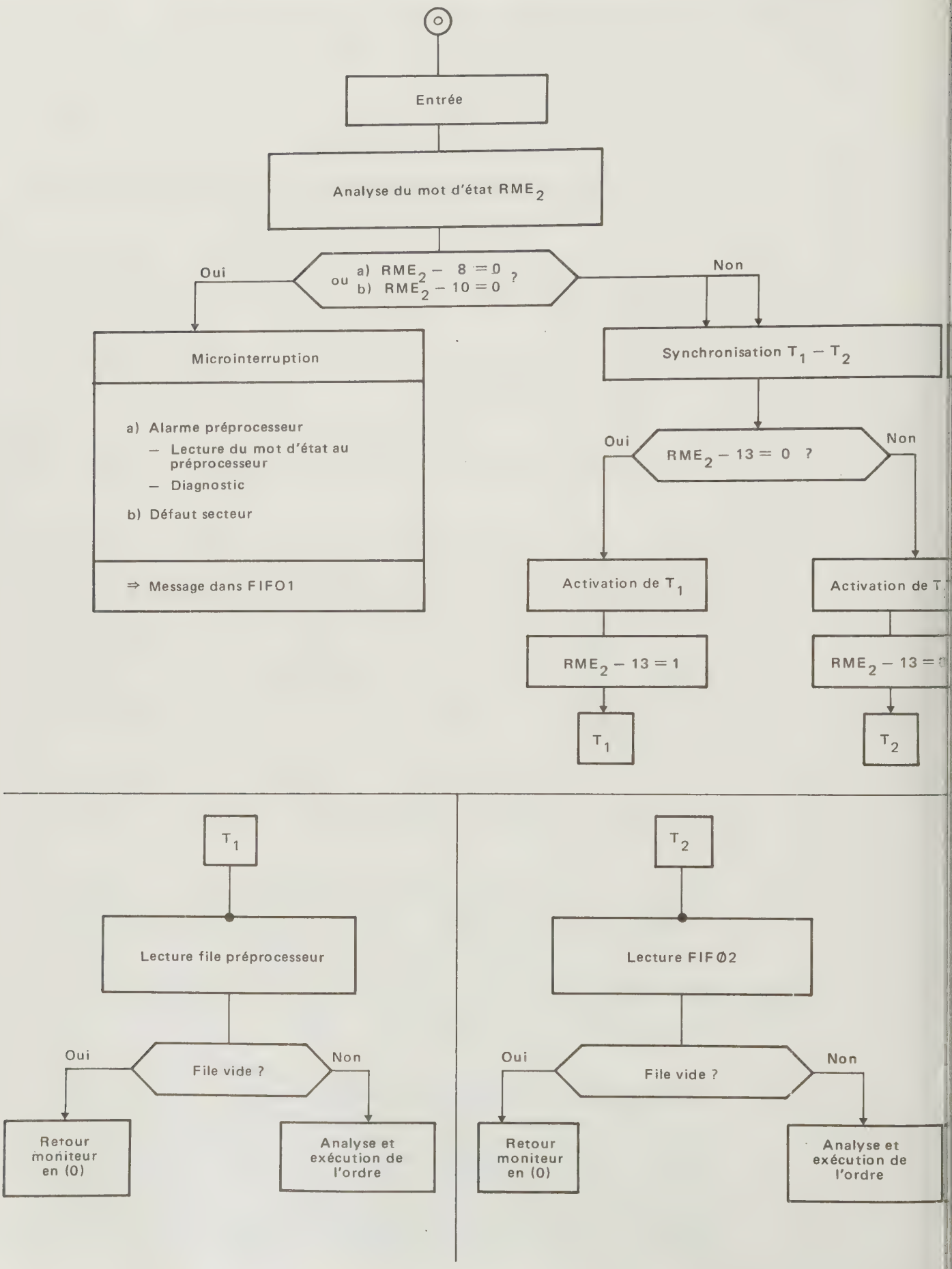
Le rôle du programme moniteur va être de synchroniser ces deux tâches de façon à suivre exactement le diagramme d'état (page précédente) ; on lui attribue également un rôle de surveillance des alarmes, c'est-à-dire que le prépocesseur peut lui envoyer un avis d'alarme au cas où un défaut est détecté dans son fonctionnement interne (débordement de la file, défaut déporté dans l'un des blocs mémoire);de même l'alimentation peut lui envoyer un avis de défaut secteur, c'est pourquoi le programme moniteur doit continuellement explorer ces différentes possibilités.

On trouvera page suivante l'organigramme du programme moniteur, celui plus détaillé des tâches T_1 et T_2 sera donné en annexe (VII).

Le moniteur est un programme qui possède un point d'accès unique, c'est-à-dire que toute tâche activée par le moniteur devra lui rendre le contrôle lorsqu'elle s'achèvera par un branchement ou un saut à ce point d'entrée. Il a à sa disposition pour l'enchaînement des différentes tâches trois drapeaux (les e.b. 8-10 et 13) du registre d'état RME_2 qu'il positionne de façon à marquer l'état suivant à atteindre.

Les éléments binaires RME_2-8 et RME_2-10 sont chargés par l'extérieur; RME_2-8 sert à la signalisation des alarmes du prépocesseur, RME_2-8 = 1 signifie qu'une faute a été détectée dans le fonctionnement interne de cet organe (défaut de parité dans l'une de ses tables, débordement de la file), RME_2-10 sert à la signalisation des défauts secteur, lorsqu'une coupure secteur survient, l'alimentation dispose d'une autonomie suffisante pour permettre à l'UMS d'en aviser l'UC:

Le programme moniteur de l'UT microprogrammée



(quelques millisecondes), RME_2-10 = 1 indique un défaut secteur. L'élément binaire RME_2-13 sert de drapeau de synchronisation pour les tâches T_1 et T_2 :

- RME_2-13 = 0 signifie que la tâche suivante à activer est T_1,

- RME_2-13 = 1 signifie que la tâche suivante à activer est T_2.

Lorsque le contrôle est revenu au moniteur, celui-ci explore séquentiellement RME_2-8 et 10, de façon à détecter une alarme éventuelle, puis considère RME_2-13.

Si RME_2-13 = 0, le programme active T_1 qui positionne aussi cet élément binaire à 1.

Si RME_2-13 = 1, T_2 est activée qui remet RME_2-13 à 0.

De cette façon, les deux tâches T_1 et T_2 s'enchaînent mutuellement et les priorités sont respectées.

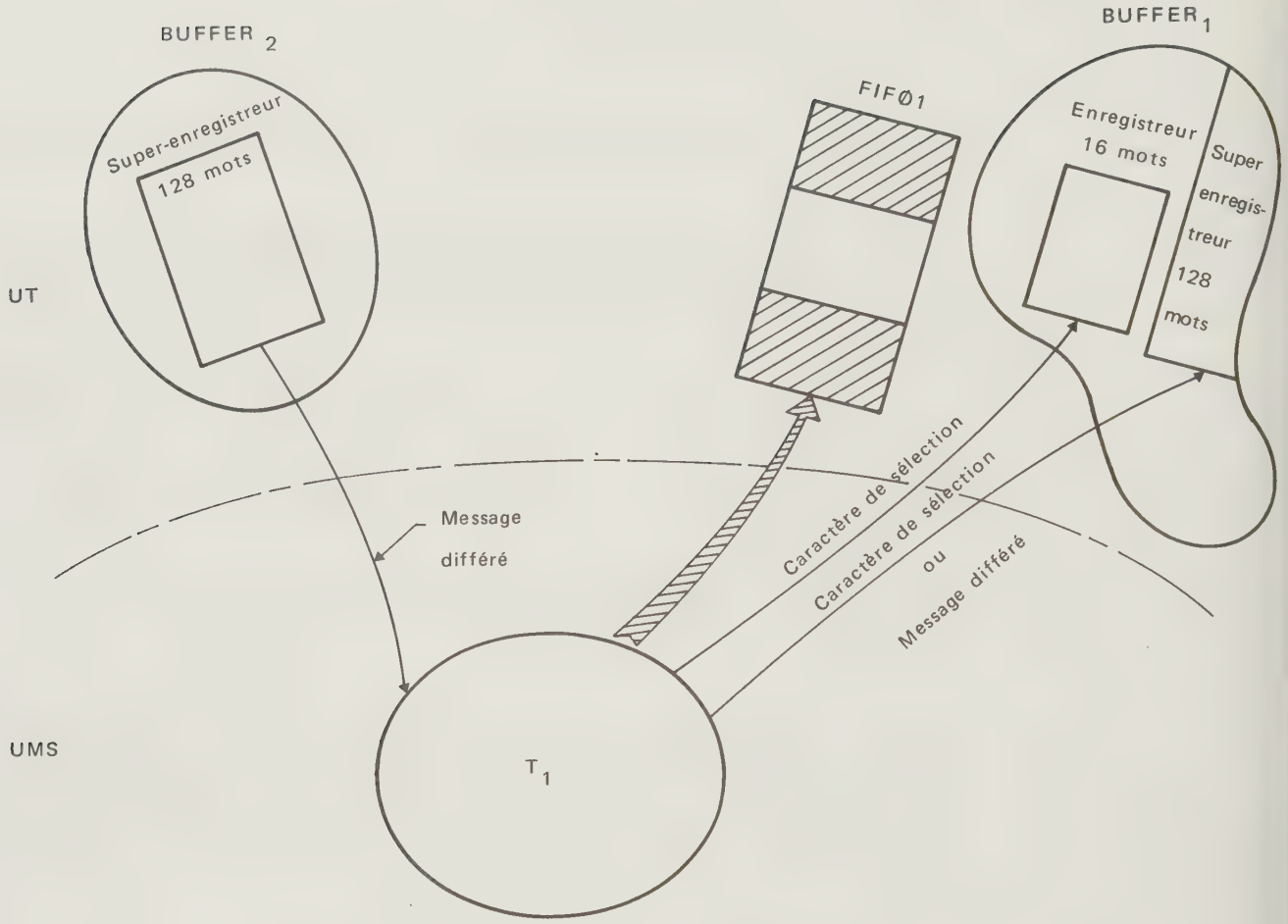
On peut encore remarquer que les deux tâches T_1 et T_2 ont exactement la même structure puisqu'elles consistent toutes deux à lire un mot dans une file d'attente, à regarder si la file n'est pas vide, et à interpréter le code de l'ordre contenu dans un champ opération du message lu.

2.3 - Les échanges avec l'unité de traitement

On a déjà vu que les échanges entre l'UMS et l'unité de traitement portaient sur des flots d'informations de nature différente, c'est-à-dire, des données reçues sur une voie (caractères de sélection ou message à diffuser en différé) ou bien des commandes.

Ces flots d'informations sont gérés par les tâches T_1 et T_2 ; nous avons vu au paragraphe précédent que la tâche T_2 ne voyait que la file d'attente FIFØ2, alors que T_1 avait accès à FIFØ1 et aux différents buffers ; nous nous intéresserons donc plus particulièrement à T_1.

Le principe qui est à la base de la gestion des différents flots est que les commandes sont échangées au moyen de file d'attente, alors que les données sont lues et (ou) écrites directement dans la mémoire vive de l'UT.



Le mécanisme de gestion des files d'attente FIFØ1 et FIFØ2 est câblé, le format des messages est défini en annexe (I) ; par contre, la gestion des zones mémoire sera à échanger, les données sont faites par programme.

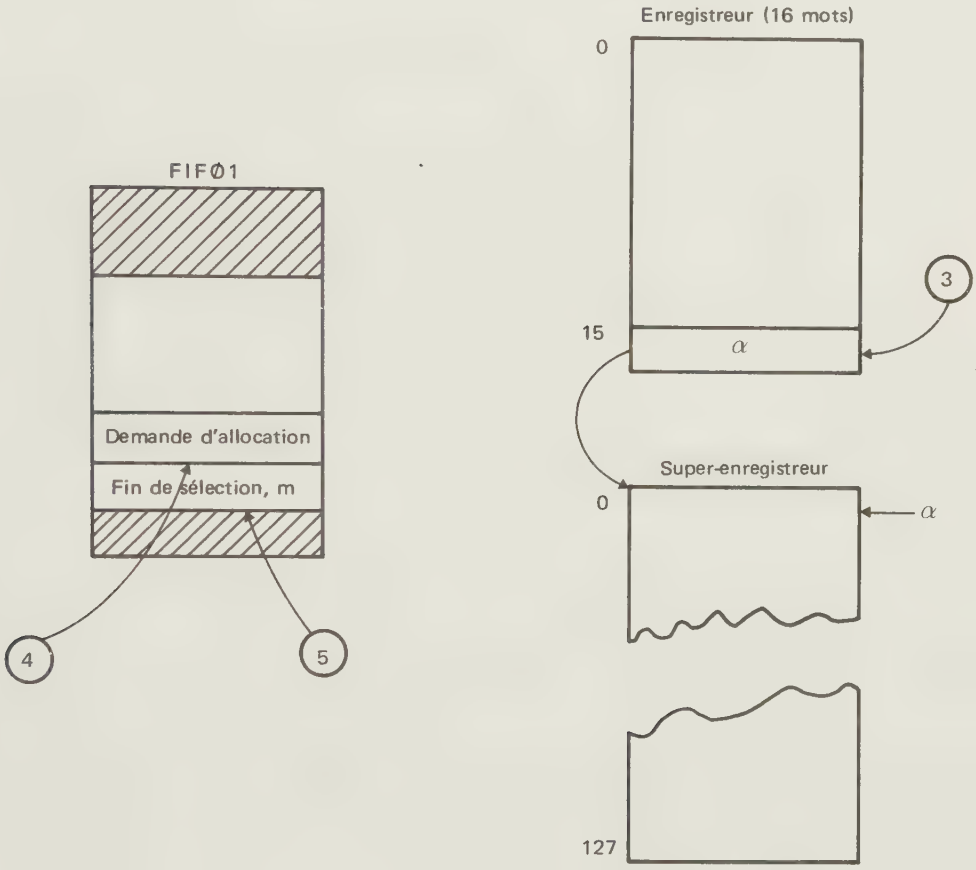
Au cours de l'établissement d'une communication, en même temps qu'elle donne l'ordre à l'UMS d'envoyer sur la voie de l'abonné appelant l'invitation à numéroter, l'UT donne à l'UMS l'adresse du premier d'une zone de 16 mots consécutifs (appelée enregistreur) dans laquelle la tâche T_1 viendra ranger les caractères de sélection reçus de l'appelant.

L'enregistreur est alors une ressource propre à l'UMS qui est seule à en disposer ; lorsque l'abonné a envoyé son caractère de fin de sélection, l'UMS libère l'enregistreur en inscrivant le message de fin de sélection dans FIFØ1, l'enregistreur est alors à la disposition de l'UT.

Ce qui précède correspond au cas normal, c'est-à-dire, au cas où 16 mots, soit 32 octets,sont suffisants pour mémoriser tous les caractères de sélection; cependant, le schéma de signalisation prévoit qu'il peut y en avoir beaucoup plus.

Pour faire face à cette éventualité, l'unité de traitement microprogrammée dispose des adresses de début de quelques zones mémoire de 128 mots chacune (super-enregistreur); ces quelques adresses sont rangées dans une pile (qui pourra être supportée physiquement par des registres de la micro-machine).

T_1 tient à jour le compte des caractères de sélection émis par l'abonné, soit n ce nombre, décomposons alors T_1 en étapes distinctes :



1) n = 30 : risque de débordement,

2) recherche d'un super-enregistreur, extraction d'une adresse de la pile, soit (α) cette adresse,

3) établissement d'un lien de chaînage, c'est-à-dire, inscription dans le mot 15 de l'enregistreur,

4) demande d'allocation d'un super-enregistreur (envoi d'un message dans FIFØ1).

Cette phase de chaînage terminée, T_1 continuera d'acquérir les caractères de sélection de l'abonné et les rangera dans le super-enregistreur à partir de l'adresse (α). La fin de sélection sera indiquée comme précédemment par :

5) Accusé de fin de sélection
(inscription dans FIFØ1).

Cet accusé de fin de sélection contient le nombre de caractères de sélection reçus, de sorte que l'UT peut déterminer,si ce nombre est supérieur à 30, l'adresse du super-enregistreur utilisé en lisant le mot 15 de l'enregistreur.

Une procédure assez semblable est utilisée pour la mémorisation et la relecture des messages transmis en différé.

2.4 - La gestion de la mémoire de travail

L'UMS dispose pour le marquage des voies en signalisation d'une mémoire de travail (MT). Dès qu'une voie initialise un échange avec l'organe de commande, une zone de quatre mots de cette mémoire (microenregistreur) lui est allouée, qui sera libérée lorsque la transaction sera terminée. Une des tâches importantes de l'UMS sera donc la gestion de cet espace mémoire.

Le microenregistreur contient, à un instant donné, toutes les informations nécessaires à l'exécution du programme correspondant à l'étape actuelle du schéma de signalisation.

Chaque tâche concernant une voie donnée est initialisée par un message provenant de la file d'échange avec le prépocesseur, l'annexe (III) décrit le format de ce message ; on y voit en particulier, qu'on peut y trouver soit l'adresse de la voie (LAV_{0-15}), soit le numéro du microenregistreur inscrit dans le mot de commande (LMC_{0-9}). En effet, lors de l'initialisation d'une transaction, la tâche T_1 doit allouer un microenregistreur à la voie;le message d'initialisation ne contiendra donc que l'adresse de la voie, pour les étapes ultérieures de la transaction. Chaque message reçu du prépocesseur contiendra le numéro du microenregistreur associé à la voie, ce qui permettra à T_1 d'accéder à l'information utile.

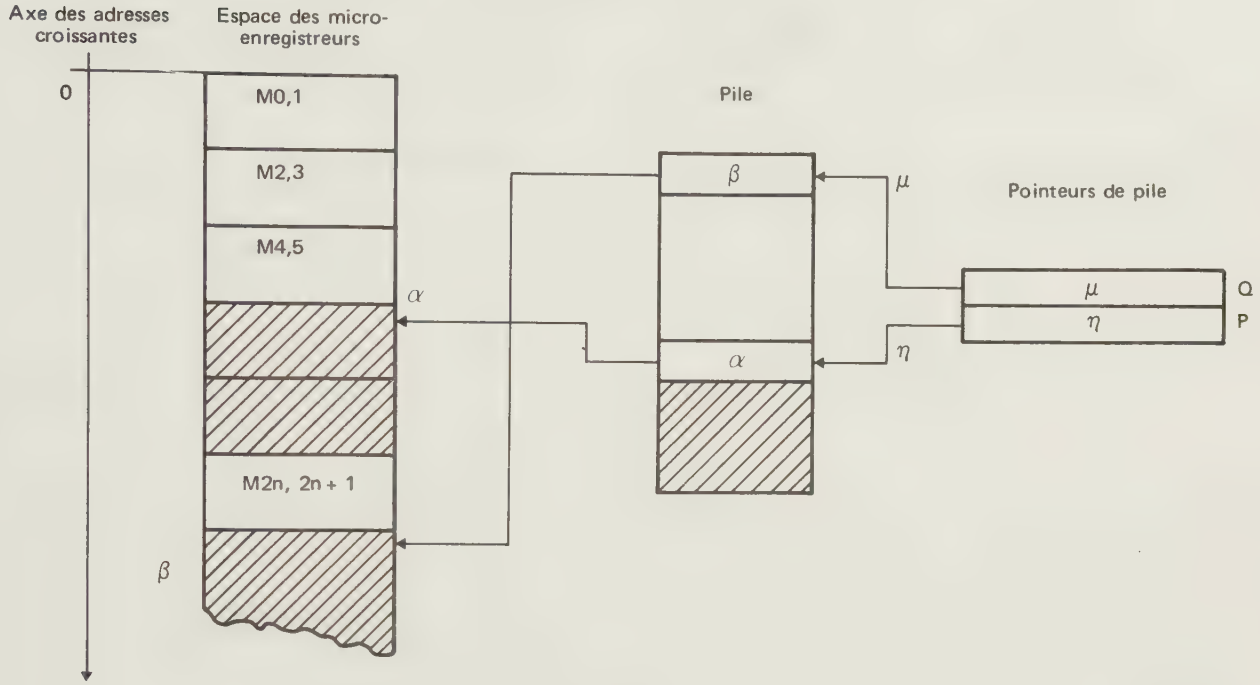
- l'allocation des microenregistreurs

L'ensemble des microenregistreurs fait partie d'une zone mémoire gérée dynamiquement au moyen d'une pile (voir figure page suivante).

La pile est définie par deux pointeurs P et Q. Q pointe sur le fond de la pile, P sur le prochain mot à extraire de la pile.

Le mécanisme d'allocation est simple ; lorsque T_1 a besoin d'un microenregistreur (ou plutôt de deux, car il est plus simple de disposer d'un double microenregistreur : l'un pour la voie du demandeur, l'autre pour la voie du demandé, utilisée ultérieure-

ment). Le programme lit les pointeurs P et Q, dans le cas général, on a $P \neq Q$, ce qui signifie qu'il y a plus d'un mot dans la pile ; dans ce cas, T_1 ne retient que P qui pointe sur le prochain mot à extraire de la pile, ce mot contient l'adresse du microenregistreur attribué, soit α .



Pour libérer un microenregistreur, il suffit de replacer son adresse dans la pile et d'incrémenter P.

Le premier mot de la pile joue un rôle particulier; en effet, il pointe sur la zone d'adresse au-delà de laquelle aucun microenregistreur n'est attribué ; ce qui revient à dire que les autres mots de la pile pointent sur des "trous" dans l'espace des microenregistreurs déjà attribués.

La gestion des microenregistreurs est faite au moyen de deux petits programmes, ou primitives, DEPIL, qui procède à l'attribution d'un double microenregistreur et EMPIL, qui libère un double microenregistreur, les organigrammes de ces deux programmes sont donnés page suivante.

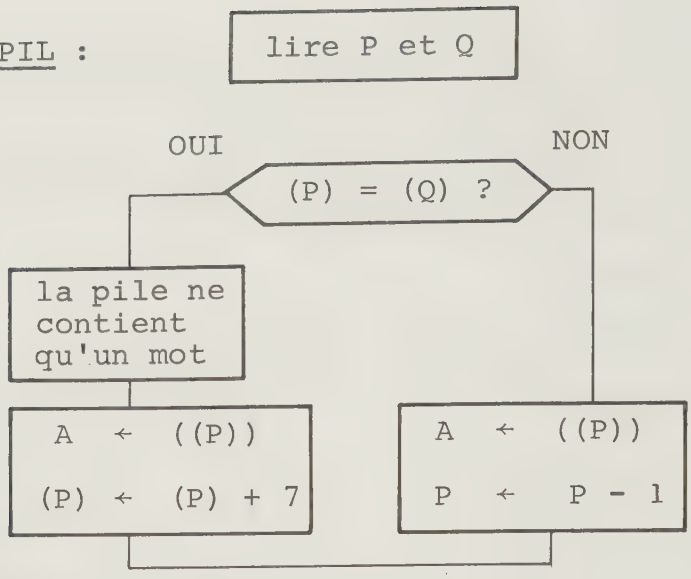
Notation : P représente l'adresse du pointeur p

(p) représente le contenu du mot d'adresse P

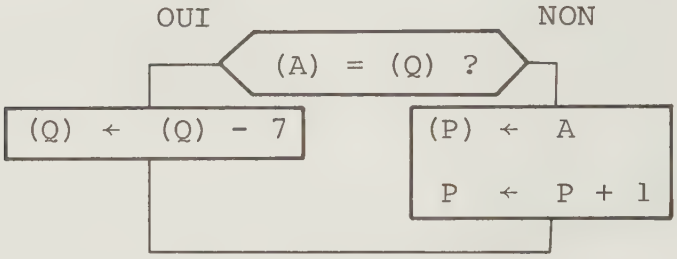
((p)) représente le contenu du mot d'adresse (p)

A représente l'adresse du mot où l'on vient ranger l'adresse du microenregistreur.

DEPIL :



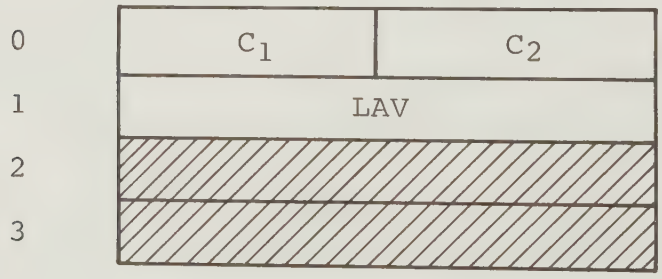
EMPIL :



- Le contenu du microenregistreur

Le microenregistreur attribué à une voie doit contenir toutes les informations nécessaires à l'exécution de la prochaine phase de traitement concernant cette voie.

Le microenregistreur comprend quatre mots :



Les deux premiers sont réservés à un usage particulier, l'utilisation des deux derniers est variable suivant la phase de traitement et le type de la voie.

Le premier mot est lu par le prépocesseur, le programme de la micromachine y inscrit deux caractères C_1 et C_2 à partir desquels le prépocesseur génère un octet (de quadrioctets) avec une phase imposée par le RCX. Cette possibilité est utilisée pour émettre sur une voie un caractère non permanent ou une chaîne de caractères, par exemple, le caractère de progression d'appel ou bien un message entier retransmis de façon

différée, si, par exemple, pour envoyer la suite de caractères, C_1, C_2, C_3, l'UT microprogrammée placera d'abord (C_1, C_2) dans le mot 0 du microenregistreur, et attendra du prépocesseur un message d'accusé de réception d'émission de l'octet correspondant. Ce message contient la phase de l'octet émis ; en fonction de celle-ci, l'UT microprogrammée détermine alors si le prochain octet émis doit être généré à partir de (C_2, C_3) ou bien encore à partir de (C_1, C_2) et modifie ou non le contenu du premier mot du microenregistreur.

Le second mot du microenregistreur contient l'adresse de la voie à laquelle il est associé, ceci est utilisé pour l'identification de la voie dans les échanges avec l'unité de traitement.

Les mots 2 et 3 contiendront, suivant le cas, le compte des caractères de sélection reçus sur une voie, le dernier caractère de sélection reçu sur la voie ou l'adresse de voie du correspondant.

2.5 - L'initialisation

Les paragraphes précédents ont eu pour but de montrer les mécanismes mis en jeu par le logiciel opérationnel de l'UMS qui est le seul nécessaire en régime de fonctionnement normal. Il faut cependant noter l'existence, parallèlement à ce processus opérationnel, d'un processus d'initialisation (ou bootstrap) inscrit dans une partie en mémoire morte de la mémoire de programme (bloc RØM).

En effet, tous les blocs mémoire de l'UMS, c'est-à-dire aussi bien les tables du prépocesseur que les blocs de mémoire de programme du processeur microprogrammé, sont technologiquement réalisés au moyen de mémoires volatiles, donc, au moment de la mise sous tension, et à chaque réinitialisation, il y a lieu de recharger ces mémoires.

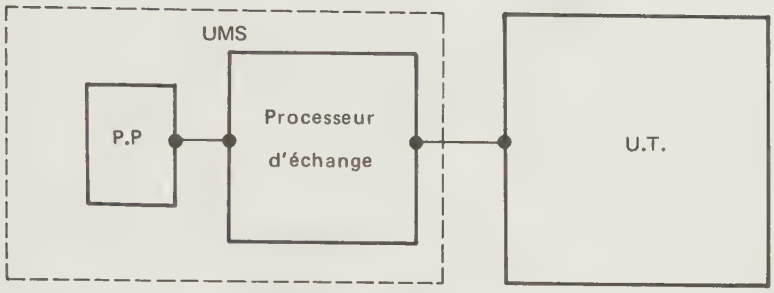
Le bootstrap est un petit programme en mémoire non volatile que l'on active au moment de la mise sous tension (ou d'une réinitialisation) par l'enfoncement de la touche "marche-programme" du pupitre de la micromachine.

Ce programme vient tout d'abord lire dans FIFØ1 l'adresse d'une zone mémoire ou sont rangés les contenus respectifs de la mémoire de programme (blocs RAM_1 et RAM_2) et des tables du prépocesseur, puis charge séquentiellement et mot à mot chacun de ces blocs.

Lorsque le changement est terminé, le programme d'initialisation lance automatiquement le programme opérationnel.

V - CONCLUSION

V.1 - La structure actuelle



La maquette de commutation est basée sur une structure de processeurs et de processus hiérarchisés. On a vu qu'une telle structure entraîne une bonne répartition de la charge suivant les trois maillons de la chaîne des données.

On a vu également que l'on supprime ainsi la notion d'interruption sauf dans l'unité de traitement où des tâches telles que la gestion des temporisations ou la gestion du disque impose l'utilisation du système d'interruptions hiérarchisées.

On peut faire à l'égard d'une telle structure deux remarques :

. L'organe de commande tel qu'il est conçu manque de fiabilité ; il suffit qu'un des maillons soit défaillant pour que l'ensemble de l'organe de commande soit dans l'impossibilité d'assurer aucun service. Cette particularité est incompatible avec les contraintes d'exploitations auxquelles doit satisfaire un système opérationnel, mais tel n'était pas notre objectif.

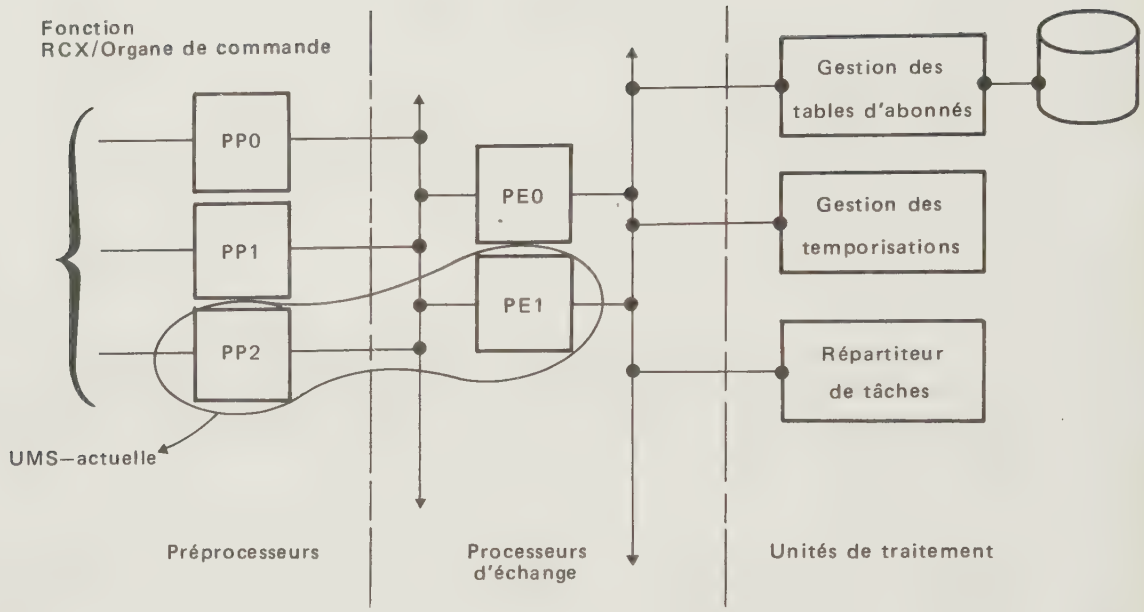
. Toutes les données reçues du RCX sont multiplexées sur un bus unique; l'organe assurant la fonction entre le RCX et l'organe de commande (le prépocesseur) doit être capable de traiter un très gros débit d'informations, il doit donc être extrêmement rapide et réalisé dans l'une des technologies les plus avancées.

Dans cette optique, nous le reconnaissons ; on peut considérer l'UMS comme un organe "coûteux", mais le multiplex unique était l'une des données qui nous était imposée.

V.2 - Un exemple de système opérationnel

La figure page suivante décrit une structure

possible pour l'organe de commande d'un commutateur qui pourrait être opérationnel.



Le principe d'une telle structure est toujours celui des processeurs hiérarchisés (3 niveaux) mais, pour chaque niveau, plusieurs processeurs sont susceptibles d'assumer parallèlement une même tâche. Ceci répond au double souci d'augmenter la fiabilité (si un processeur est défaillant, le système continuera de fonctionner) et de réduire le coût de chaque processeur (chacun ne prend en compte normalement qu'une part du trafic et n'aura pas besoin d'être très performant).

Les voies entrantes, au lieu d'être multiplexées sur un bus unique, sont réparties selon plusieurs multiplex de niveau inférieur. De ce fait, le prépocesseur qui n'a plus besoin d'être si rapide peut être réalisé plus simplement dans une technologie moins délicate à mettre en oeuvre et moins coûteuse.

Dans ce cas de figure, un prépocesseur traite un multiplex ; si ce prépocesseur est défaillant, il s'ensuit une détérioration limitée au service ; seules les voies du multiplex vues par ce prépocesseur et leurs voies associées sont touchées par cette pertubation.

Les processeurs microprogrammés (unités d'échange) sont également multipliés ; ils fonctionnent en par-

tage de trafic ; il suffit donc d'en connecter suffisamment, donc de diminuer les contraintes sur chacun d'eux pour les faire entrer dans la gamme des petits processeurs du commerce.

Le dernier étage surtout doit subir des modifications par rapport à la maquette. Pour supprimer la notion d'interruption, on peut éclater l'UT en autant de processeurs spécialisés que la fonction commutation nécessite de catégories de tâches différentes. En première approximation, on peut convenir que trois classes de processeurs sont suffisantes :

. Des processeurs d'enchaînement de tâches, qui jouent le rôle du programme moniteur de l'UT,

. Des processeurs de gestion des tables d'abonnés,

. Des processeurs de gestion des temporisations.

Tous les processus s'exécutent en parallèles et communiquent entre eux au moyen de files d'attente.

Un nouveau problème apparaît dans ce schéma du fait que le chemin de données ne peut être unique. En effet, lorsqu'un prépocesseur détecte une demande de service, il inscrit dans sa file d'attente, un message vers un processeur d'échange qu'il ne distingue pas.

Lorsqu'un processeur d'échange a du temps de libre, il explore cycliquement les nouvelles demandes de services en provenance des prépocesseurs, il en choisit une et il traitera la transaction jusqu'au bout.

Il devra lui-même adresser dans une phase ultérieure une demande de service aux processeurs d'enchaînement de tâches et se verra attribuer pour cette transaction le moins chargé :le chemin de données ainsi constitué reste fixe pendant toute la durée de la transaction.

V.3 - Conclusion

L'exemple ci-dessus est une anticipation mais il pourrait réellement représenter la structure d'un commutateur de données opérationnel offrant un service fiable.

Une telle réalisation n'était pas notre but, il nous fallait seulement faire la preuve de la viabilité d'une nouvelle technique. Toutefois, on peut remarquer que l'UMS (prépocesseur et processeur d'échange) peut être considérée comme un sous-ensemble d'une structure de type parallèle (voir figure).

On peut enfin remarquer que le passage au stade de la maquette à celui du modèle définitif, n'implique en fait que des modifications du matériel. Quelle que soit la structure choisie, les mécanismes logiciels décrits dans ce document demeurent applicables puisqu'ils consistent es-

sentiellement à permettre les échanges et la synchronisation de processus parallèles, les fonctions réalisées par ces processus étant de toute façon nécessaires.

o

o o

BIBLIOGRAPHIE

(1) Les Progrès de la Commutation Electronique dans le monde
P. LUCAS
Commutation Electronique n°44, janvier 1974

(2) Transmissions Numériques
F. PLATET, H. COURANT, J-P. PRIGENT, Y. MADEC
L'Echo des Recherches, n°71, janvier 1973

(3) Spécifications Techniques et Logiques concernant l'Interface entre le Réseau de Connexion du commutateur Hermes et son Organe de commande
G. LELARGE
Documentation CNET/ITD/TDP/25 FTM 0423
juin 1974

(4) Additif à la Spécification Technique ITD/TDP/25 pour harmoniser le Matériel et le Logiciel dans la Maquette du Commutateur Hermes

(5) Principe de base des Systèmes de Commutation Temporelle intégrée - Structure et Synchronisation des Réseaux- Commut. Electr.
Colloque international PARIS (28 mars-2 avril 1966)
A. PINET

(6) Les Problèmes de Réseaux de Connexion en Commutation Temporelle
F. TALLEGAS
Commutation et Electronique, n°23 - Octobre 1968

(7) Proposition pour l'Architecture du Réseau Hermes
Note Technique, projet IITD/TDP/5
4 juillet 1972

(8) Définition des Expériences menées à l'aide de la Maquette de Commutateur Hermes
ITD/TDP/25
septembre 1974

(9) Etat d'Avancement du Réseau Hermes
Note Technique, projet ITD/TDP/21
mai 1974

(10) Maquette Hermes
l'Unité de Marquage et de Signalisation du Commutateur de Données Hermes
par G. LELARGE
Note Technique, projet ITD/TDP/22
avril 1974

(11) L'Unité de Marquage et de Signalisation de la Maquette de Commutateur Hermes
G. LELARGE
Note Technique, projet ITD/TDP/28
mars 1975

(12) Réseau de Commutation Numérique du type à Multicadence par Multiplex
Mm TEXIER et LAPEYRONIE
Brevet 74 43 116

(13) Micro programming : An Introduction and a view point
M.J. FLYNN et R.F. ROSIN
IEEE Transactions on Computers
Vol. C-20, n°7 - juillet 1971

(14) Application of the Théory of Automata to Minimization of Microprograms
V.M. GLUSHKAR
Engineering Cybernetics - n°1 - janvier-février 1964

(15) Livre vert du CCITT - Tome VIII (1972)
Transmissions de données

(16) Synchronization of communicating processes
A-N. HABERMANN
CACM, Vol.15, n°3
mars 1972

(17) Switching and Finite Automata théory
par A. KOHAVI
New-York, Mc GRAW-Hill, 1970

(18) Cahier des Charges des Autocommutateurs Telex
Dossier K.3130 CNET-RCC-CRA

(19) Définition d'un Microassembleur pour le Calculateur T 1600 Télémécanique
Note Technique, projet
J-P. GUENIN

A N N E X E S

ANNEXE I

-:-:-

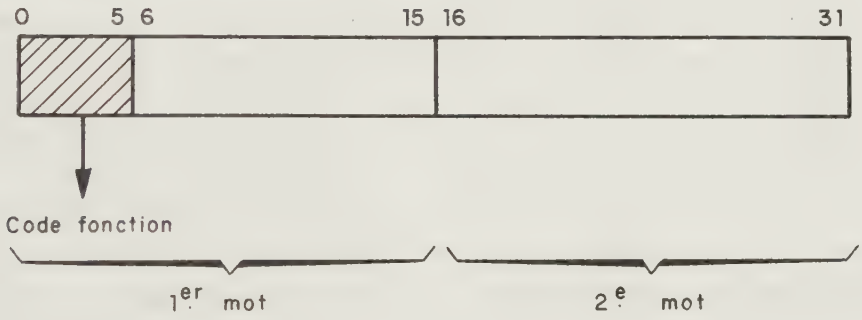
FORMAT DES MESSAGES ECHANGES ENTRE UMS ET UT

-:-:-

1. - GENERALITES

Le format du message élémentaire est de 32 e.b. La taille du mot de la mémoire vive de l'UT étant de 16 e.b., un message sera rangé dans l'une des files d'attente sous forme de deux mots mémoire.

D'une façon générale chaque message comporte un code fonction sur 6 e.b. (d'où 64 possibilités) qui prennent la place des e.b. 0 à 5 du premier mot du message lu ou écrit.



Le jeu de messages défini, ci-après répond particulièrement à la gestion de la procédure de signalisation définie spécialement pour la maquette (qui est une approximation de l'avis X21) il convient également à la gestion de la procédure Télex [18] et il peut être aisément étendu lors de l'adjonction de nouveaux services.

2. - SENS UMS VERS UT

Le deuxième mot lu (e.b. 16 à 30) contiendra systématiquement l'adresse de la voie sur laquelle porte la commande.

Désignation	Code fonction (binaire)	Signification
Demande d'appel	0 0 0 0 0 1	Un abonné vient d'émettre une demande d'appel - un microenregistreur lui a été attribué dont le numéro est codé dans le champ des e.b. 6 à 14 au premier mot.
Fin de sélection	0 0 0 0 1 0	L'abonné vient de terminer sa sélection. Il vient d'envoyer n caractères que l'UT trouvera dans l'enregistreur affecté à cet abonné et éventuellement dans un super enregistreur. n est codé dans le champ des e.b. 6 à 15 du premier mot.
Confirmation d'appel	0 0 0 0 1 1	L'abonné à qui le commutateur envoyait de façon continue un signal d'arrivée d'appel vient de répondre en émetteur un caractère de confirmation d'appel.
Accusé de connexion	0 0 0 1 0 0	Ce message est envoyé de l'UMS vers l'UT après exécution d'un ordre de connexion de deux abonnés initialisé par celle-ci. Lors de l'émission de ce message, les deux abonnés sont déjà en communication. L'adresse de voie (e.b. 16 à 30) correspond à celle de l'abonné demandeur.
Demande de libération	0 0 0 1 0 1	Une demande de libération vient d'être reçue sur la voie considérée. L'UMS vient de lui affecter un microenregistreur, dont le numéro est codé dans le champ d'e.b. 6 à 15, elle est marquée mais reste dans l'état de communication.
Accusé de libération	0 0 0 1 1 0	Ce message fait suite à un ordre de libération lancé par l'UT, il signifie que l'un des abonnés vient d'émettre un accusé de réception de libération.

Demande de libération avec rappel immédiat	0 0 0 1 1 1	L'abonné considéré vient d'émettre une demande libération mais qui sera suivie immédiatement de la demande d'un nouveau service. Un microenregistreur vient d'être affecté à la voie qui sera réutilisée pour le traitement de la nouvelle demande, l'UT ne devra pas pour les mêmes raisons libérer son minienregistreur.
Alarme de voie	0 0 1 0 0 0	Le préprocesseur vient de détecter avec alarme sur la voie dont l'adresse est donnée dans le champ e.b. 16 à 30. Elle doit être immédiatement mise en état de faute par l'UT, et l'ordre doit en être transmis à l'UMS.
Alarme de muldex	0 0 1 0 0 1	Le préprocesseur vient de détecter une alarme concernant le muldex dont le numéro est codé dans le champ (e.b. 16 à 30), chacune des voies de ce muldex devra être mise en faute par l'UT, un ordre de mise en faute devra être donné à l'UMS pour chacune des voies.
Demande de changement d'état d'une voie	0 0 1 0 1 0	Un abonné vient d'effectuer une demande de changement d'état. L'état demandé est codé dans le champ d'e.b. 6 à 14. Le code des différents états possibles est donné ci-dessous (notation binaire). Ligne libre : 0 1 Mode local : 1 0 Hors service : 1 1 Lors de l'émission de ce message l'UMS n'a pas encore modifie le mot d'état de la voie - elle en attend l'ordre de l'UT.
Alarme UMS	0 0 1 0 1 1	Ce message ne comporte pas d'autre indication que le code fonction il signifie que l'UMS est hors d'état d'assurer son service.

Accusé de connexion générale	0 0 1 1 0 0	L'abonné a initialisé une procédure de diffusion, l'UT a précédemment envoyé à l'UMS, l'ordre de mise en connexion générale, c'est-à-dire de permettre au demandeur (celui dont l'adresse de voie est codée dans ce message) de commencer à émettre. L'ordre donné par l'UT vient d'être exécuté.
Accusé de fin de réception d'un message demandeur	0 0 1 1 0 1	L'abonné considéré a demandé une diffusion différée d'un message. Ce message vient d'être émis et inscrit dans un super-enregistreur (buffer) de la mémoire de l'UT.
Accusé de fin d'émission d'un message différé	0 0 1 1 1 0	L'UT a précédemment demandé à l'UMS de transmettre à un abonné (celui dont la voie est codée dans le champ e.b. 16 à 30) un message différé, ceci a été exécuté et une demande de libération envoyée à l'abonné considéré. L'UT peut disposer du Buffer pour un autre usage et doit attendre la confirmation de libération de l'abonné.
Demande d'allocation d'un super-enregistreur	0 0 1 1 1 1	L'UMS dispose en permanence d'un jeu de quelques super-enregistreurs (buffers de 128 mots chacun dans la mémoire vive de l'UT) qui lui servent à mémoriser les caractères de sélection d'un abonné lorsque leur nombre excède la capacité d'un simple enregistreur. Pour faire face à toute éventualité l'UMS doit toujours en avoir à sa disposition et peut pour celà en faire une demande à l'UT qui lui en alloue un à chaque fois.

2. - SENS UT VERS UMS

Dans ce cas la structure du message est moins rigide, elle sera rappelée pour chaque fonction.

	Désignation	Code fonction (binaire)	Signification
0	File vide	0 0 0 0 0 0	Ce message n'est pas une commande de l'UT, il est généré par le système de gestion de la file d'attente (ADM) pour indiquer à l'UMS que la file est vide. Il ne comporte aucune autre information utile.
1	Invitation à numéroter	0 0 0 0 0 1	C'est un ordre donné à l'UMS d'envoyer sur la voie spécifiée dans le champ des e.b. 16 à 30, le caractère d'invitation à numéroter (ce caractère sera envoyé de façon continue jusqu'à la réception du caractère de fin de solution. Le champ d'e.b. 6 à 15 contient le numéro du micro enregistreur associé à la voie.
2	Accusé de réception du caractère de fin de sélection.	0 0 0 0 1 0	Ordre donné à l'UMS d'émettre sur la voie spécifiée dans le champ des e.b. 16 à 30 un caractère permanent d'accusé de réception du signal de fin de sélection. Le champ d'e.b. 6 à 15 contient le numéro du microenregistreur associé à la voie.
3	Avis d'arrivée d'appel	0 0 0 0 1 1	Ordre donné à l'UMS d'émettre sur la voie spécifiée dans le champ d'e.b. 16 à 30 le caractère permanent ayant la signification "arrivée d'appel". Le champ d'e.b. 6 à 15 contient le numéro du microenregistreur associé à la voie.
4	Progression d'appel	0 0 0 1 0 0	Ordre donné à l'UMS d'émettre une seule fois le caractère de progression d'appel codé dans le champ d'e.b. 16 à 23, sur la voie associée au microenregistreur dont le numéro est codé dans le champ d'e.b. 6 à 15.

5	Connexion	0 0 0 1 0 1	Ordre de connexion (cas de la communication simple entre deu abonnés). • le message contient le numéro du microenregistreur associé à la communication (e.b. 6 à 15) et l'adresse de la voie du demandé (e.b. 16 à 30). L'UMS doit mettre les deux abonnés en état de communication inscrire dans le mot de commanc de l'un l'adresse de voie de l'au et réciproquement, libérer le microenregistreur et renvoyer un accusé de connexion à l'UT.
6	Libération	0 0 0 1 1 0	Cet ordre fait suite à une deman de libération d'un abonné. L'adr se de la voie de cet abonné est inscrite dans le microenregistreu dont le numéro est codé dans le champ d'e.b. 6 à 15 et l'adre se de voie de son correspondant est lu dans le champ d'e.b. 16 à 30. L'UMS entamera la procédure de libération et libèrera le microenregistreur lorsque les de voies seront revenues à l'état ligne libre, un accusé de réceptic de libération sera envoyé à l'UT à chacune de ces transitions.
7	Changement d'état d'un abonné	0 0 0 1 1 1	Le message comporte l'adresse de voie de l'abonné (e.b. 16 à 30) et le nouvel état de la voie (e.b. 6 à 14) codé comme suit : Ligne libre : 0 1 Mode locale : 1 0 Hors service : 1 1 Réseau hors service : 1 0 0
8	Connexion diffusion récepteur	0 0 1 0 0 0	Cet ordre est utilisé dans le cas de la diffusion d'un message émis par un abonné vers n autres.

			L'un de ces n abonnés a reçu un signal d'arrivée d'appel et envoie de façon permanente un signal de confirmation. L'UT vient de décider de le faire passer dans l'état de réception-diffusion. La liaison ainsi établie est unilatérale. Les caractères émis par l'abonné diffuseur transiteront par l'UMS avant de parvenir aux récepteurs. L'adresse de la voie de l'abonné est codée dans le message (e.b. 16 à 30).
9	Connexion diffusion Emetteur	0 0 1 0 0 1	Lorsqu'un abonné demande la diffusion d'un message vers n autres l'UT met d'abord ces n abonnés dans l'état connexion diffusion récepteur puis donne l'ordre à l'UMS de permettre au demandeur d'émettre son message. Ce message comprend l'adresse de voie du demandeur (e.b. 16 à 30) et le numéro du microenregistreur associé à l'opération (e.b. 6 à 15).
10	Libération-diffusion Récepteur	0 0 1 0 1 0	Lorsque la diffusion d'un message est terminée l'UT donne l'ordre à l'UMS de libérer chacun des récepteurs. Le message de libération comprend l'adresse de voie de l'abonné à libérer (e.b. 16 à 30).
11	Libération d'un micro-enregistreur	0 0 1 0 1 1	L'UT peut à tout moment déclencher une procédure de défense qui consiste en la libération de ressources indûment mobilisées. Ainsi l'UT peut demander la libération d'un micro enregistreur qu'elle désigne par son numéro (e.b. 6 à 15).

12	Allocation d'un super enregistreur	0 0 1 1 0 0	Ce message est consécutif à une demande d'allocation d'un super enregistreur formulée par l'UMS. Le super enregistreur est désigné par son numéro (e.b. 6 à 15), l'algorithme qui permet à partir du numéro de déterminer l'adresse au premier mot de la zône mémoire correspondante, est défini au moment de l'initialisation.
13	Emission différée d'un message	0 0 1 1 0 1	Dans lecas de remise différée d'un message l'UMS doit connaître l'adresse de voie de l'abonné vers qui elle doit envoye le message (e.b. 16 à 30) et le numéro du micro enregistreur associé à l'opération (e.b. 6 à 15) dans lequel elle a préalablement inscrit l'adresse au début de la zône mémoire contenant le message.
14	Connexion différée	0 0 1 1 1 0	Lorsqu'un abonné demande le service de remise différée d'un message l'UT lui alloue une zône mémoire pour y inscrire son message. La mise en connexion consiste dans ce cas à demander à l'UMS de lire ce message et l'inscrire dans la zône mémoire. Puisque la voie de demandeur est déjà marquée il suffit à l'UMS de connaître le numéro de son micr enregistreur (e.b. 6 à 15). l' adresse du début de la zône mémoire peut être donnée dans le champ d'e.b. 16 à 30.

15	Mémorisation d'appel	0 0 1 1 1 1	Lorsqu'un abonné demande un appel avec mémorisation éventuelle et que son correspondant n'est pas libre, on place le demandeur dans un état d'appel mémorisé et l'on conserve le micro enregistreur qui lui était réservé. Le message comprend le numéro du micro enregistreur (e.b. à 15) et l'adresse de voie au futur correspondant (e.b. 15 à 30).

ANNEXE II

-:-:-

PROCEDURE DE SIGNALISATION TERMINALE

-:-:-

1. - Signalisation à l'interface d'abonné :

1.1. - Généralités :

a) Code :

Tous les signaux sont des caractères de l'IA5 avec bit d'imparité, sauf 0 (0000 0000) et $\hat{I}$ (11111111).

b) Utilisation des fils de contrôle :

Le fil "contrôle" est à l'état OFF pendant les états permanents du terminal (Repos, Local, Hors Service...). Il devient ON à la demande d'appel et redevient OFF à la libération ou bien au rappel immédiat du commutateur.

Le fil "Indication" est ON pendant la phase "données", OFF le reste du temps.

1.2. - Etats permanents :

	DE	DR	Contrôle	Indication
Repos (ligne libre)	$\hat{0}$	$\hat{0}$	OFF	OFF
Mode local	1/11	-	OFF	-
Hors service	$\hat{1}$	-	OFF	-
Réseau hors service	-	$\hat{1}$	-	OFF

1.3. - Sélection :

1.3.1. - Forme générale :

- Début d'en tête : & (2/6)
- Service spécial : 1 ou 2 caractères alphanumériques
- Séparateur de services : , (2/12)
- Fin d'en tête : / (2/15)
- Adresse : 1 à 12 caractères alphanumériques

- Séparateur d'adresses : , (2/12)
- Fin de sélection : + (2/11)
- Caractère de bourrage : SYN (1/6)

1.3.2. - Forme simplifiée : appel sans service spécial :

- Adresse : 1 à 12 caractères alphanumériques
- Séparateur : , (2/12)
- Fin de sélection : + (2/11)

S'il y a plusieurs adresses, il y a éventuellement autant de caractères de progression d'appel que d'adresses.

Si l'appel ne comporte qu'une adresse il n'y a évidemment pas de séparateur.

1.3.3. - Adresse :

a) Adresse complète :

Pour la communication locale : 1 à 5 chiffres.

b) Adresse abrégée :

Elle est définie par l'utilisateur et comporte 1 ou 2 chiffres (100 numéros par l'utilisateur).
Elle s'utilise précédée du préfixe A comme une adresse normale : Exemple :

A12 équivalent à 51982.

1.3.4. - Services spéciaux :

a) Diffusion :

S'obtient directement par la présence de plusieurs adresses. La communication est unilatérale (demandeur vers demandé). Il y a autant de caractères de progression d'appel que d'adresses.

b) Numérotation abrégée :

Pour définir des numéros abrégés :
&0/numéro abrégé, adresse complète, num. ab., ad. comp., ...+
numéro abrégé : 1 ou 2 chiffres.
Les adresses abrégées ainsi définies s'utilisent n'importe où, en remplacement d'une adresse complète, précédées du préfixe A.

Exemple : après la commande &0/7, 21513 + A7 est équivalent à 21513.

c) Création d'un réseau fermé :

Une station maîtresse, si sa discrimination le permet, peut placer une ou plusieurs stations en état d'esclaves. C'est-à-dire que ces stations deviennent inaccessibles à tout autre utilisateur et qu'elles fonctionnent en appel direct (hot-line) vers la station maîtresse. Simultanément la station maîtresse définit des numéros abrégés pour ses esclaves.

& 1/numéro abrégé, adresse complète, num. ab., ad. comp., ...+

d) Affranchissement des esclaves :

Permet de rendre sa liberté à une station définie comme esclave. &2/adresse, ad.,...+

e) Transfert d'appel :

Permet de transférer les appels arrivants vers une autre station. Un seul transfert est admis.

& 3/adresse +

f) Annulation d'un transfert :

& 4 +

g) Remise différée :

Le commutateur enregistre le message et le distribue ensuite à toutes les stations adressées, lorsque celles-ci deviennent libres.

& 5/adresse, ad.,... +

Cette requête est suivie d'une connexion et le commutateur enregistre toutes les données émises par le terminal (sauf "SYN") jusqu'à l'émission de deux C.R. consécutifs.

S'il n'y a pas d'erreurs le commutateur répond "OK" et libère.

S'il y a des erreurs le commutateur répond "ER" et le terminal peut recommencer.

h) Mémorisation de l'appel :

Si l'appel n'aboutit pas immédiatement le commutateur l'enregistre et rappellera demandeur et demandé lorsque les deux seront libres. Cette procédure n'est possible qu'avec un seul demandé.

& 6/adresse +

1.4. - Caractère de progression d'appel :

Il est constitué d'un caractère alphanumérique.

0 demandé libre

1 demandé occupé

2 demandé hors-service

3 demandé en mode local

4 réseau incapable d'acheminer l'appel

5 erreur de parité

6 syntaxe de la sélection incorrecte

7 requête interdite au demandeur

8 requête satisfaite (lorsque celle-ci ne conduit pas à une connexion)

Le caractère 0 est suivi d'une connexion.

Les caractères 1 à 8 sont suivis d'une libération par le réseau.

1.5. - Collision d'appel :

Le terminal choisit d'imposer son appel ou d'accepter l'appel arrivant.

Pour accepter l'appel arrivant il répond normalement (contrôle = ON ; DE = "ACK").

Pour imposer son appel il maintient sa demande d'appel (Contrôle = ON ; DE = "SYN") jusqu'à réception de l'invitation à numéroter.

1.6. - Rappel immédiat :

Afin d'abréger la signalisation un terminal peut libérer une communication en faisant simultanément une nouvelle demande d'appel.

Pour cela, au lieu d'émettre le signal de libération (OFF - $\hat{0}$) il émet OFF - "ETX" (0/3) et le commutateur répond directement par l'invitation à numéroter ("+") ; le terminal commence la sélection avec contrôle ON.

En dehors de ce cas particulier, un terminal ne peut demander un appel qu'à partir de l'état "ligne libre".

1.7. - Appel direct (Hot-line)

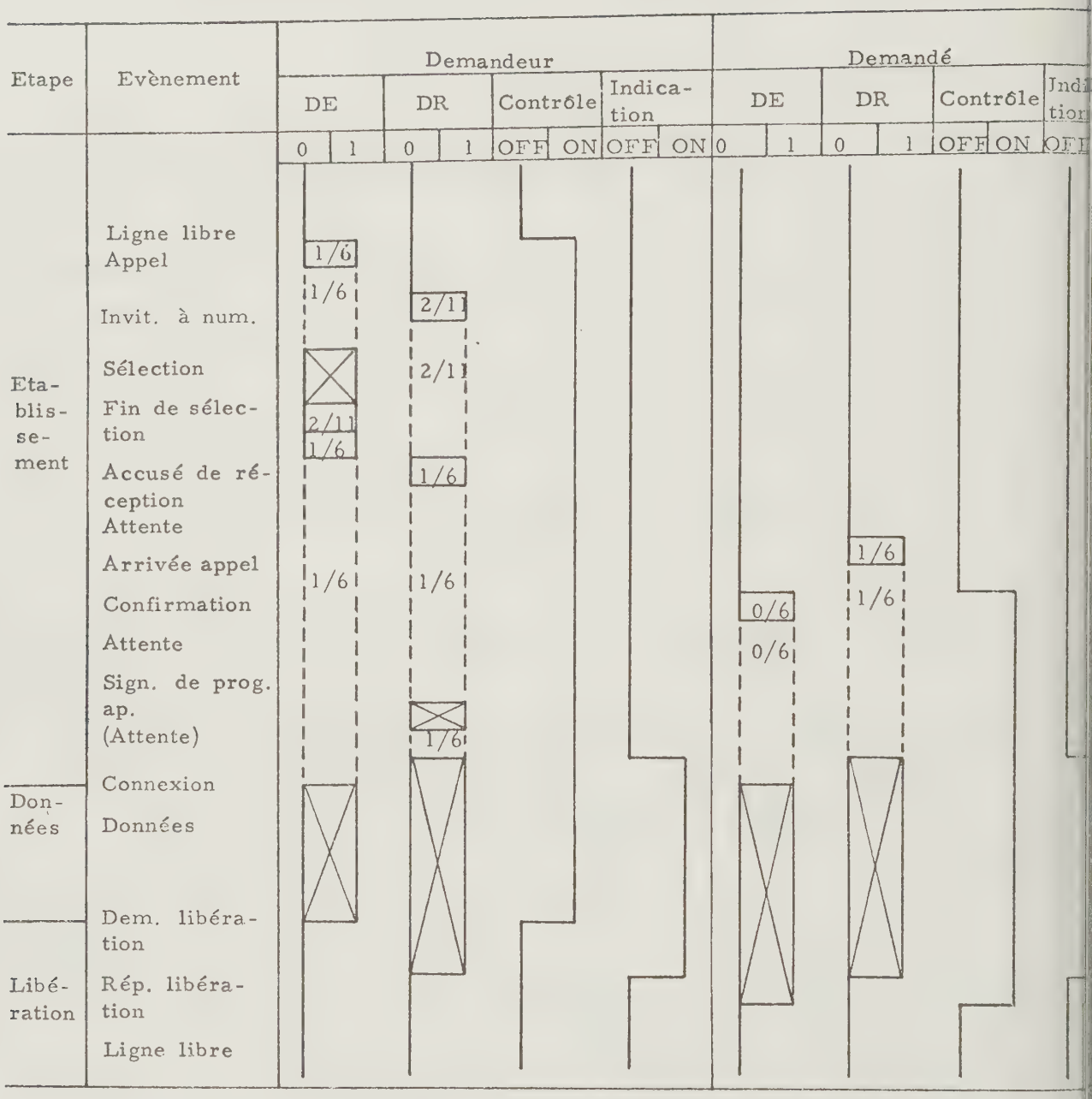
Cette discrimination ne permet à un terminal d'obtenir qu'un seul correspondant. Sa demande d'appel est directement suivie du signal de progression d'appel (facultatif) et du signal de connexion s'il a lieu.

1.8. - Délai de garde :

Responsable	Début	Fin	Durée en caractères	Action à l'expiration du délai
Terminal	Demande d'appel	Invitation à numéroter	500	Libération
Réseau	Invitation à numéroter	Fin de sélection	5000	Libération
Terminal	Fin de sélection	Signal de connexion ou de libération	500	Libération
Réseau	Libération par le réseau	Libération par le terminal	50	Mise en état hors service du terminal
Terminal	Libération par le terminal	Libération par le réseau	500	Néant
Réseau	Arrivée appel	Confirmation	50	Libération sauf en cas de collision d'appel : prise en compte de l'appel
Terminal	Confirmation	Connexion	500	Libération
Terminal	Libération	Demande d'appel	50 (Mini)	Le terminal est autorisé à lancer un nouvel appel

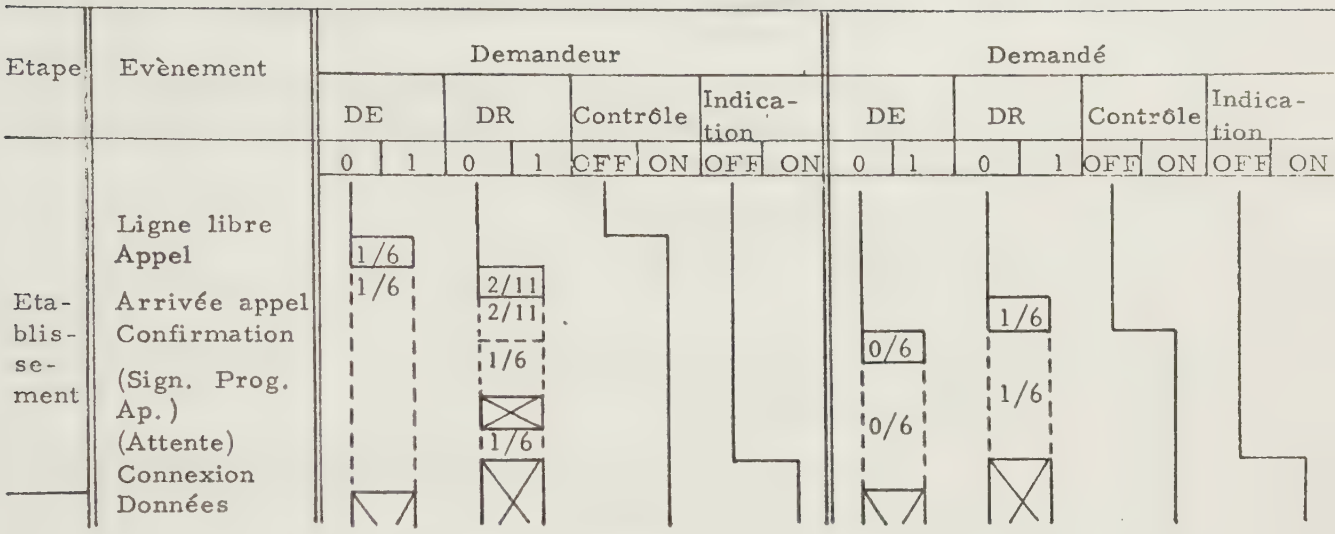
1.9. - Diagramme de signalisation

a) Appel avec sélection :

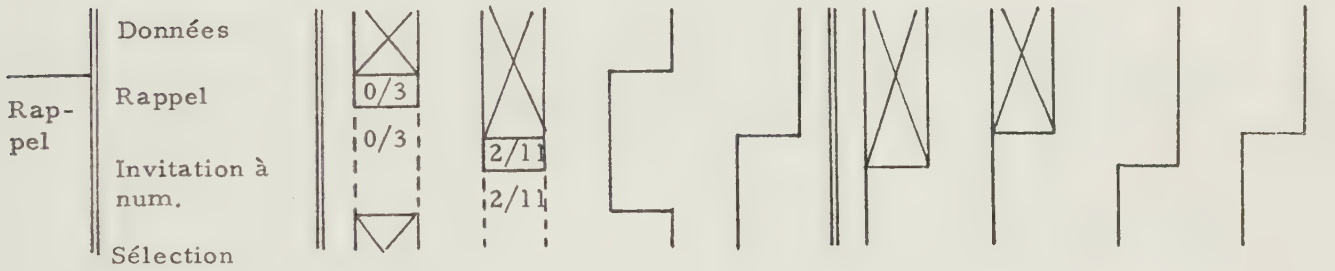


- Les pointillés indiquent des caractères répétés un nombre quelconque de frais (éventuellement zéro).
- Le diagramme représente le cas où le demandeur demande aussi la libération.

b) Appel direct (hot-line) :



c) Rappel immédiat :



1.10. - Signal de connexion

Il est défini par la transition du fil indication de OFF à ON.

Il signifie que toutes les données reçues après lui proviennent du correspondant et que toutes les données émises après lui seront reçues par le correspondant après réception du signal de connexion.

Les premières données reçues après le signal peuvent avoir été émises par le correspondant avant la réception du signal de connexion.

k) Convention logique :

OFF = 1 ON = 0

2. - Description de l'interface électrique :

a) Liste des circuits :

L'interface comporte 9 circuits affectés aux broches d'un connecteur 25 points.

Terre de protection	broche 1	
Emission de données	broche 2	
Réception de données	broche 3	
Terre de signalisation (réception)	broche 7	
Indication	broche 12	(ETD)
Contrôle	broche 11	
Horloge bit	broche 15	
Horloge caractère	broche 21	
Terre de signalisation (émission)	broche 25	

Le connecteur est de type mâle du côté du terminal.

b) Caractéristiques électriques :

Conformes au document ISO 796.

Emetteur : Impédance interne : Ri 50

Tension différentielle en circuit ouvert : 2 V 3

Courant de court-circuit : Icc 150 mA

Tension sur 50 : V 1 V

Courant pour 0,5 V sur les sorties, alimentation coupée : I 100 µA

Temps de montée (0,1 à 0,9 V) : tr 0,1 x temps bit

Récepteur : Impédance d'entrée 4 K

c) Convention logique :

ON et 0 logique + 2,5 volts

sur le fil actif

OFF et 1 logique - 2,5 volts

d) Diagramme des temps :

Horloge bit :	ON
	OFF
	ON
Horloge caractère	OFF
	ON
Contrôle, indica-	OFF
tion	ON
Données	OFF

Les circuits Contrôle et Indication sont échantillonnés respectivement par l'ETL et le Terminal sur la transition ON à OFF de l'horloge caractère. Leur valeur à cet instant est relative au caractère dont le premier bit est transmis à cet instant.

ANNEXE III

-:-:-:-

1. - LE BLOC PRIMAIRE

T	CAT_{0-7}	HS	CAR1	NLAV	CE_{0-7}	$MP1_{0-7}$	CAR2	$MP2_{0-7}$	C1
0	1 2 3 4 5 6 7 8	9	10	11	12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27	28	29 30 31 32 33 34 35 36	37

T : Inhibition au test du caractère entrant ;

T = 0 inhibition du test du caractère entrant (CR_{0-7})

T = 1 test du caractère entrant.

CAT_{0-7} ; caractère attendu ;

CAT est un caractère du code CCITT n° 5.

HS : type du traitement choisi (si le caractère reçu est le caractère attendu) ;

HS = 0 traitement "Hardware" fonction câblée dans le préprocesseur.

HS = 1 traitement par programme du niveau de l'UMS.

CAR_1 : Localisation du caractère à émettre (si le caractère reçu est le caractère attendu) ;

CAR_1 = 0 le caractère à émettre est CE_1 considéré comme caractère permanent.

CAR_1 = 1 le caractère à émettre est dans MT (adresse dans LMC).

NLAV : Inscription dans la FIFO de l'adresse de voie ;

NLAV = 0 inscription de LAV dans la FIFO.

NLAV = 1 inscription de PH_{0-1} et de l'adresse du micro enregistreur dans la FIFO.

CE : Caractère permanent à émettre (Si $CAR_1 = 0$) ;

CE est un caractère du code CCITT n° 5.

MP_1 : Type du module à lancer pour l'UMS (si le caractère attendu est le caractère reçu).

$0 \leqslant (MP_1) \leqslant 255$

CAR_2 : Localisation au caractère à émettre (cas où le caractè attendu n'est pas le caractère reçu).

$CAR_2 = 0$ le caractère à émettre est CE_1.

$CAR_2 = 1$ le caractère à émettre est dans MT (adresse dans LMC).

MP_2 : Type de module à lancer par l'UMS ;

(cas où le caractère reçu n'est pas le caractère attendu) ou bien lien de chainage vers le bloc secondaire.

$0 \leqslant (MP_2) \leqslant 255$

C_1 : Chainage vers le bloc secondaire ;

$C_1 = 0$ chainage vers le bloc secondaire

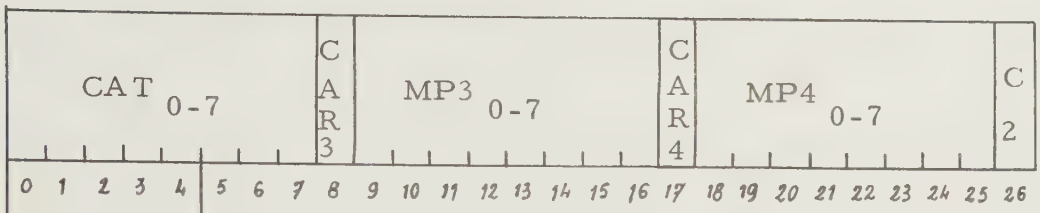
$C_1 = 1$ pas de chainage (MP_2 est une information à destination de l'UMS)

Remarque : Le super filtrage c'est-à-dire le refus par le préprocesseur de tou nouvel appel (tout en continuant la gestion des transactions déjà engagées) est activé par la combinaison.

$$\begin{cases} T = 0 \\ HS = 0 \end{cases}$$

Cette combinaison n'avait aucun sens dans le cas général.

2. - LE BLOC SECONDAIRE



CAT : Caractère attendu ;

CAT appartient à l'alphabet CCITT n° 5.

CAR_3 : Localisation du caractère à émettre (cas où le caractère attendu est le caractère reçu)

$CAR_3 = 0$ le caractère à émettre est CE_1.

$CAR_3 = 1$ le caractère à émettre est dans MT.

MP_3 : Type du module à lancer par l'UMS (si le caractère reçu est le caractère attendu).

$0 \leqslant (MP_3) \leqslant 255$

CAR_4 : Localisation du caractère à émettre (cas où le caractère reçu n'est pas le caractère attendu).

$CAR_4 = 0$ le caractère à émettre est CE_1.

$CAR_4 = 0$ le caractère à émettre est dans MT.

MP_4 : Type du module à lancer par l'UMS

(Si le caractère attendu n'est pas le caractère reçu) ou bien lien de chainage dans le bloc secondaire.

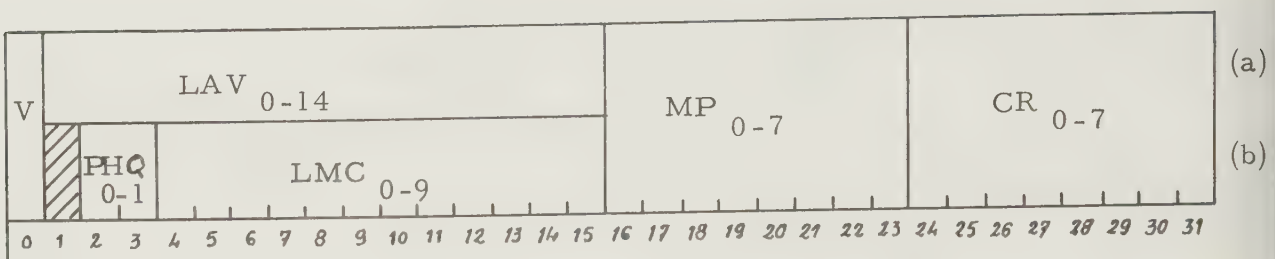
$0 \leqslant (MP_4) \leqslant 255$

C_2 : Chainage dans le bloc secondaire ;

$C_2 = 0$ chainage dans le bloc secondaire (l'adresse est dans MP_4).

$C_2 = 1$ pas de chainage (MP_4 est une information à destination de l'UMS).

3. - LA FILE D'ATTENTE CABLEE



V : Indicateur de remplissage de la FIFO ;

V n'est pas un élément binaire mémorisé dans la file ; il est généré par la logique de gestion de la FIFO et permet de faire savoir à l'UMS si la FIFO est vide (le mot reçu est alors sans signification.

V = 0 - la file est vide

V = 1 - le mot reçu est bien significatif.

MP_{0-7} : Type du module à lancer par l'UMS ;
MP peut être MP_1, MP_2, MP_3 ou MP_4.

CR_{0-7} : Caractère reçu ;
(voir note ITD/TDP/25).

a) LAV_{0-14} : Adresse de la voie considérée ;
(voir note ITD/TDP/25).

b) PH_{0-1} : Phase de l'octet émis au cours de l'IT présent ;
(voir note ITD/TDP/25).

LMC_{0-9} : Numéro du microenregistreur attribué à la voie trait
(voir note ITD/TDP/25)

Lors de l'initialisation d'une transaction entre un abonné et l'organe de commande le mot inscrit dans la FIFO contiendra l'adresse de la voie (cas a) de façon que l'UMS lui réserve un micro enregistreur dont le numéro sera inscrit dans le mot de commande (LMC_{0-9}).

ANNEXE IV

-:-:-

NOTICE D'UTILISATION DU GENERATEUR DE TABLE DE DECISION

-:-:-

Le langage de commande

(*) émis par GENTADE précède la frappe de toute commande.

1 * CREER (RC)

BO : nom FIC_0 (RC)

BT : nom FIC_1 (RC)

BP : nom FIC_2 (RC)

BS : nom FIC_3 (RC)

(nom FIC_I) = 6 caractères alphanumériques au max.

crée et ouvre sur le disque (fu D4) sous le catalogue LG les fichiers (nom fic_i)

La création de tout fichier est facultative.

BO correspond au fichier binaire séquentiel.

(binaire objet) qui sera récupéré ultérieurement et chargé dans les tables de décision.

BT, BP, BS correspondent aux fichiers symboliques archivés.

GENTADE émet après chaque ligne reçue le caractère (-), il suffit de frapper (RC) pour voir apparaître (*).

<u>NB</u> : Gentade émet un message d'erreur si la création est impossible.

2 (*) FIN (RC)

Ferme tous les fichiers créés au cours de la vacation et rend le contrôle au superviseur.

3 (*) GENERER (RC)

(-) BT : C_0, C_1, C_2, C_3 (RC)

C_0 : = T ou NT type de RCX

C_1 : = S ou NS bit S

C_2 : = C ou NC communication ou non.

C_3 : = un chiffre de 0 à 31.

si C_2 = "C" C_3 est non significatif (facultatif).

BT génère un lien de chainage vers un mot du bloc primaire (dans fichier provisoire en mémoire centrale).

En cas d'erreur de syntaxe ou de logique la commande est ineffective et un message d'erreur est affiché.

(-) BP : C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9 (RC)

C_0 : = T ou NT inhibition de la comparaison.

C_1 : = CAT (= "caractère α" ou "UN" ou "ZE" ou "SYN")

"UN" génère un tout 1

"ZE" génère un tout 0

"SYN" génère un caractère SYN

C_2 : = HS ou NHS

C_3 : = CAR ou NCAR

C_4 : = LAV ou NLAV

C_5 : = CE caractère émis (même convention que pour CA

C_6 : = MP_1 (n° de prog).

MP_1 = xx (deux chiffres hexa.).

C_7 : = CAR ou NCAR

C_8 : = MP_2 (n° de prog ou adresse de chainage des BS) même convention que pour MP_1

C_9 : = C ou NC.

Les blancs ne sont pas significatifs. La ligne doit se terminer par (CRC).

BP génère le binaire correspondant à la ligne symbolique et l'inscrit dans une table à l'adresse pointée par la ou les ligne générées par la dernière commande BT.

En cas d'erreur de syntaxe ou de logique gentade émet un message d'erreur. En cas de chainage vers le BS BP génère lui-même le lien de chainage (en cas d'impossibilité - BS plein - Gentade émet un message d'erreur).

<u>NB</u> : L'adresse zéro de BP contient une configuration particulière (traitement d'erreur) de façon que toute configuration non recon pointe sur elle.

(-) BS : C_0, C_1, C_2, C_3, C_4, C_5 (RC).

C_0 : = CAT

C_1 : = CAR ou NCAR

C_2 : = MP_3

C_3 : = CAR ou NCAR

C_4 : = MP_4

C_5 : = C ou NC

Les conventions sont les mêmes que pour BP.

4 * INIT

Réinitialise le système le fichier temporaire BT est remis à zéro, tous les pointeurs sont réinitialisés.

Les fichiers préalablement créés ne sont ni fermés ni détruits.

5 (*) DETRUIRE (RC)

(-) BO (RC)

(-) BT (RC)

(-) BP (RC)

(-) BS (RC)

Détruit au choix les fichiers BO, BT, BP, BS créés au cours de la vacation et donne un compte rendu d'éxécution.

6 (*) ASSEMBLER (RC)

Génère à partir des fichiers binaires temporaires le fichier BO définitif créé sous le nom nom FIC_0 par la commande CREER.

ASSEMBLER donne un listage en hexa du contenu de BO.

7 (*) LISTER (RC)

- BT (RC)
- BP (RC)
- BS (RC)

Liste sur l'unité symbolique LO le contenu du fichier désigné.

ANNEXE V

SYNOPTIQUE DU MOT DE LA MICROINSTRUCTION

M	0 1 2 3 4	5 6	7 8	9	10 11	12 13 14 15 16 17 18 19	20 21 22 23 24 25 26 27	28 29	30 31
CAS GENERAL	ADRESSE DE REGISTRE 00000 REGISTRES GENERAUX (R00) 01111 RME 10 = 0 JEU UT (R15) RME 10 = 1 JEU μ CANAL 10000 AUCUN (PAR) 10001 RTR (RTR) 10010 RCPT (CPT) 10011 10100 TAP DROIT (TAD) 10101 TAP GAUCHE (TAG) 10110 RDEL (RDE) 10111 REX (REX) 11000 AP SANS SAUVEGARDE (RUP) 11001 AP AVEC SAUVEGARDE (ASP) 11010 RP (RPO) 11011 RC (RAD) 11100 RAD1 ORIGINE MOD DESTIN. (AD1. MOD) 11101 RAD2 ORIGINE MOD DESTIN. (AD2. MOG) 11110 RME1 (RM1) 11111 RME2 (RM2)	OPERATION 00 UNION (UNI) 01 DISJONCTION (DIS) 10 INTERSECTION (INT) 11 ADDITION (ADD)	REPORT INITIAL 00 ZERO ( ) 01 UN (RI1) 10 $RME1_{15}$ (RIR) (CARRY) 11 $L1\emptyset_0$ (RIB) REMARQUE : $L1\emptyset_0$ INTERDIT AVEC L'ADDITION	POSITIONNEMENT DE CARRY ET OVERFLOW 0 NON ( ) 1 OUI (PRE) $RME1_{15}$ - CARRY $RME1_{16}$ = OVERFLOW	POSITIONNEMENT DE T 00 RIEN ( ) 01 TEST (TNU) NUL 10 TEST (TNE) NEG 11 TEST (TPO) POSITIF 1ER CAS : VALEUR ET M17Ø CAL. DEC. E/S ET M17 = 0 00 INCHANGE 01 SI LR . 0 10 SI LR < 0 11 SI LR > 0 2EME CAS CAL. DEC. E/S ET M17 = 1 00 FORCE A1 01 SI LR ≠ 0 10 SI LR ≥ 0 11 SI LR ≤ 0		SI M29 = 0 COMMANDES RR MEMOIRE (M00 A M15) / TRAIT. N° DE BIT / EMPLOI DE RMOD M00 = RIEN M01 = LECRE AP + 1 M02 = LECRE AP M03 = LECRE AP + Δ M04 = LEBOF AC + Δ M05 = LEBOF AC + 1 M06 = LEBOF AC + CST M07 = (LEC)RE M08 = (LEC)RE PUIS M01 M09 = (LEC)RE PUIS M02 M10 = (LEC)RE PUIS M05 M11 = INUTILE M12 = (EF)ECR M13 = (EF)ECR PUIS M01 M14 = (EF)ECR PUIS M05 RIEN 00 RIEN ( ) 01 RAZ DE LR D'APRES CPT (ZBC) 10 N° DE BIT DS CPT (BCD) 11 INTERDIT 0 NON ( ) 1 OUI (MAI) REMARQUE - UNION DE (RMOD) ET M2. 3. 4. UNIQUEMENT POUR RG. SI M29 = 1 ADRESSE DE RUPTURE (ETIQUETTE VALEUR DECIMALE ± VALEUR DECIMALE)	ADRESSE μ INSTRUC. SUIVANTE 00 SEQUENCE 10 RETOUR DE SOUS PROGRAMME 01 T=0 SEQ T=1 RUP 11 T=0 RETOUR SS PROG. T=1 RUP CF : REMARQUE 3	CODE D'ORDRE
CALCUL		CAS GENERAL	CAS GENERAL $L1\emptyset_0$ EXCLUS	CAS GENERAL	CAS GENERAL 1ER OU 2EME SUIVANT M17	REGISTRE DESTINATAIRE - TOUS SAUF AP - MODIF TEST 0 ( ) 1 (TAI) CF-M10 ET 11 REGISTRE 2EME OPER. 00 ZERO 01 REX (DRE) 10 RTR (TRN) 11 RTR (TRB)	CAS GENERAL	CAS GENERAL	00 (CAL)
DECALAGE	REGISTRE 1ERE OPERANDE ($L1\emptyset_0$) POSSIBLES : RG, RTR, CPT, RME_1, RME_2, PC, DEL, AD1, AD2	00 1 PAS A DROITE (DD1) 01 1 PAS A GAUCHE (DG1) 10 8 PAS A DROITE (DD8) 11 8 PAS A GAUCHE (DG8)	CAS GENERAL $L1\emptyset_0$ PERMIS NE JOUE QUE SUR LES DECALAGES D' 1 PAS	CAS GENERAL	CAS GENERAL 1ER OU 2EME SUIVANT M17	REGISTRE DESTINATAIRE MODIF TEST 0 ( ) 1 (TAI) CF-M10 ET 11 REGISTRE 2EME OPER. 00 ZERO 01 REX (DRE) 10 RTR (TRN) 11 RTR (TRB)	CAS GENERAL	CAS GENERAL	10 (DEC)
E/S		CAS GENERAL	CAS GENERAL $L1\emptyset_0$ EXCLUS	CAS GENERAL	CAS GENERAL 1ER OU 2EME SUIVANT M17	FONCTION ESCLAVE Si M19 = 0 (F00 A F15) 0 ENTREE (ENT) 1 SORTIE (SOR) MODIF TEST 0 ( ) 1 (TAI) CF-M10 ET 11 0 REG DEST. SL ( ) 1 REG OR ET DEST (YOP) CF REM. ① 0 PAS DE 2EME OPER ( ) 1 2EME OPER = RTR (YOP) CF REM ① 0 ADR ET FCT D'APRES LA INSTRUC. ( ) 1 ADR ET FCT ESCLAVE PRIS DS RCPT (IND) CF REM ②	RIEN ADRESSE ESCLAVE SI M19 = 0 (E00 A E15) TRAITEMENT N° DE BIT (CAS GENERAL) EMPLOI DE RMOD (CAS GENE)	00 SEQUENCE 10 RETOUR SS PROG. 01 SEQUENCE QQ SOIT T 11 T = 0 RETOUR SS PROG T = 1 SEQUENCE	11 (ESO)
VALEUR	REGISTRE ORIGINE MEME CHOIX QUE CI-DESSUS. REGISTRE SEULEMENT DESTINATAIRE (M7. M8 = 11) TOUS REGISTRES POSSIBLES MEME AP.	CAS GENERAL	10 (REG)*VAL → REG (ROD) 00 (REG)* VAL (ROS) 11 ZERO* VAL → REG (RDS) 01 (REG)* VAL → RTR (RO7)	CAS GENERAL	CAS GENERAL 1ER CAS SEULEMENT QQ SOIT M17	VALEUR	CAS GENERAL	CAS GENERAL	01 (VAL)

REMARQUE ①

EN ENTREE : M 18 = 0 ZERO*REX→REG
INFOBUS <
M 18 = 1 (REG)*REX→REG (YOP)

EN SORTIE : M 18 = 0 (REG)*ZERO→REG
→ INFOBUS

REMARQUE ②

	ADRESSE ESCLAVE	FONCTION ESCLAVE
M 19=0 ( )	M21 A 24	M 12 A 15
M 19=1	RCPT	RCPT -

REMARQUE ③

- POUR AVOIR M28 = 1, CODE SYMBOL : (RSP)
- POUR AVOIR M29 = 1, IL SUFFIT DE DONNER UNE ADRESSE DE RUPTURE. DANS μ INSTRUCTION E/S.
POUR M29 = 1, IL FAUT AUSSI CODER UNE ADRESSE DE RUPTURE : (+ 1)

ANNEXE VI

LISTE DU MICRO-ASSEMBLEUR

```
0267T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR EPRINT

10C
20C
30C
40C PROGRAMME ANASYN
50C ANALYSE SYNTAXIQUE
60C+SUBROUTINE DES DIFFERENTES PSEUDO INSTRUCTIONS
70C
80C
90C PROGRAMME D ANALYSE SYNTAXIQUE DES INSTRUCTIONS
100
110       INTEGER OSYM,OBIR,OBIA,OTSYM
120       INTEGER  ETIQ1,ETIQ2,ALF(132),PTVIRG,VIRG,BLANC
130       INTEGER ALFES
140       INTEGER LV
150       INTEGER SEPAG,SEPAD,QUOTE,BLANC4,CARSUIV,NODER,NUM(17),NB(6)
160       INTEGER PLUS,MOINS,STAR,FIN,CPTORD
170       INTEGER L(16),BET(132),ERR,NPROG1,NPROG2,VALEUR
180       INTEGER A,NUMINS,ERREUR(16)
190       DATA A/4HA   /
200       INTEGER VBIN,VDEC
210       INTEGER PLASYM(100,3),ALFA(132),TABSYM(50,6)
220        INTEGER ORDRE(4)
230       DATA (ORDRE(IU),IU=1,4)
240      & /4HFSYM ,4HFBIA,4HFBIR,4HFTAS /
250       INTEGER NOMMAC(2),TMACRP(10,4),TMACRU(10,4),PMACRP,
260      &PMACRO,FMACRO(10,32),FMACRP(10,32),CPTORM,BINREL(50,32)
270       INTEGER TAIPRO(50,4)
280       DATA PLUS/4H+   /,MOINS/4H-   /
290       INTEGER UN,ZERO,VPD,VALD,SYMB1,SYMB2,TYSYM,VBIN,VDEC
300       INTEGER VAR(16),CH(16,2),FCC(4),FCD(4),FCBIN(4)
310        INTEGER REGEN(16),REGBIN(16),RCD1(11),RCD1B(11),RCD2(15)
320       INTEGER RCD2B(15),RVAL(16),RVALB(16),RESO(14),RESOB(14),FE(16)
330       INTEGER FEBIN(16),ES(2),RU(4),RUB(4),COB(4),ESB(2),DEC,CAL,VAL,ESO
340       INTEGER VALB(8)
350       INTEGER VAR2(11)
360       DATA UN,ZERO /4H1   ,4H0   /
370       DATA(COB(I),I=1,4)/2H00,2H01,2H10,2H11/
380       DATA CAL,DEC,VAL,ESO /3HCAL,3HDEC,3HVAL,3HESO/
390       DATA(REGEN(I),I=1,16) /4HR00 ,4HR01 ,4HR02 ,4HR03 ,4HR04 ,4HR05 ,
400      &4HR06 ,4HR07 ,4HR08 ,4HR09 ,4HR10 ,4HR11 ,4HR12 ,4HR13 ,4HR14 ,4HR
410       DATA(REGBIN(I),I=1,16)/4H0000,4H0001,4H0010,4H0011,
420      &4H0100,4H0101,4H0110,4H0111,4H1000,4H1001,4H1010,
430      &4H1011,4H1100,4H1101,4H1110,4H1111/
440       INTEGER REGBIN2
450       DATA REGBIN2/1H0/
460       DATA (RCD1(I),I=1,10) /4HPAR ,4HRTR ,4HCPT ,4HRM1 ,4HRM2 ,4HRPO ,
470      &4HRDE ,4HRAD ,4HAD1 ,4HAD2 /
480       DATA (RCD1B(I),I=1,10)/4H0000 ,4H0001 ,4H0010 ,4H1110 ,
490      &4H1111 ,4H1010 ,4H0110 ,4H1011 ,4H1100 ,4H1101 /
500       DATA (FCC(I),I=1,4) /4HUNI ,4HDIS ,4HINT ,4HADD /
510       DATA (FCBIN(I),I=1,4)/2H00,2H01,2H10,2H11/
520       DATA (FCD(I),I=1,4) /4HDD1 ,4HDG1 ,4HDD8 ,4HDG8 /
```

```
      DATA(FS(I),I=1,2)/4HENT ,4HSOR /
      DATA(ESB(I),I=1,2)/1H0,1H1/
      DATA  (RCD2(I),I=1,13)/4HRTR ,4HCPT ,4HTAD ,4HTAG ,4HREX ,4HPAR ,
     &4HMOD ,4HMOG ,4HRM1 ,4HRM2 ,4HRDE ,4HRPO ,4HRAD  /
      DATA (RCD2B(I),I=1,13)/ 4H0001 ,4H0010 ,4H0100 ,4H0101 ,
     &4H0111 ,4H0000 ,4H1100 ,4H1101 ,4H1110 ,4H1111 ,4H0110 ,
     &4H1010 ,4H1011 /
      INTEGER CHI(10),ALPH(26)
      DATA CHI(9)/1H9/,CHI(10)/1H0/
      DATA ALPH(1)/1HA/,ALPH(26)/1HZ/
      DATA (RVAL(I),I=1,15)/4HPAR ,4HRTR ,4HCPT ,4HTAD ,4HTAG ,4HREX ,
     &4HASP ,4HRDP ,4HRM1 ,4HRM2 ,4HMOD ,4HMOG ,4HRPO ,4HRAD ,4HRDE /
      DATA (RVALB(I),I=1,15)/4H0000 ,4H0001 ,4H0010 ,4H0100 ,4H0101
     &,4H0111 ,4H1000 ,4H1001 ,4H1110 ,4H1111 ,4H1100 ,4H1101 ,4H1010
     &,4H1011 ,4H0110 /
      INTEGER TYPE,TYPO(8)
      DATA (TYPO(I),I=1,8)/4HPROG,4HFINP,4HDEF ,4HREF ,4HPNOM,
     &4HREFM,4HEGAL,4HPEND /
      INTEGERDO(4),DOB(4),RI(4),RIB(4),RP(2),RPB(2),PT(3),PTB(4)
      INTEGER ALFI(132),ALFAO,BETAO
      DATA ALFAO/1H0/
      BETAO=ALFAO-(ALFAO/2**27)*2**27
      ALFAO=ALFAO-BETAO
      INTEGER FT,FTB(2),SP(2),SPB(2),RV(2),RVB(2),AS(2),ASB(2)
      INTEGER AE(16),AEB(16),GB(16),GBB(16),BC(4),BCB(4),MO(2),MOB(2)
      INTEGER VALDEC
      INTEGER AS1,AS1B(2)
      DATA (DO(LLL),LLL=1,4)/4HNDO  ,4HDRE  ,4HTRN  ,4HTRB  /
      DATA (DOB(I),I=1,4)/2H00,2H01,2H10,2H11/
      DATA (RI(LLL),LLL=1,4)/4HRI0  ,4HRI1  ,4HRIR  ,4HRIB  /
      DATA (RIB(I),I=1,4)/2H00,2H01,2H10,2H11/
      DATA (RP(LLL),LLL=1,2)/ 4HPRE ,4HNRE /
      DATA RPB(1)/1H1/,RPB(2)/1H0/
      DATA (PT (LLL),LLL=1,3)        /4HTNU  ,4HTNE  ,4HTPO  /
      DATA (PTB(LLL),LLL=1,4)/2H01,2H10,2H11,2H00/
      DATA  FT /4HTA1  /
      DATA(FTB(LLL),LLL=1,2) /1H0,1H1/
      DATA (SP(LLL),LLL=1,2) /4HESP  ,4HNSP  /
      DATA (RVB(LLL),LLL=1,2)/1H1,1H0/
      DATA (RV(LLL),LLL=1,2) /4HYOP  ,4HNOP  /
      DATA(SPB(IL),IL=1,2)/1H1,1H0/
      DATA (ASB(LLI),LLI=1,2)/1H0,1H1/
      DATA (AE(LLL),LLL=1,16)/4HE00  ,4HE01  ,4HE02  ,4HE03  ,4HE04  ,4HE05
     &4HE06  ,4HE07  ,4HE08  ,4HE09  ,4HE10  ,4HE11  ,4HE12  ,4HE13  ,
     &4HE14  ,4HE15  /
      DATA (AEB(LLL),LLL=1,16)/4H0000,4H0001,4H0010,4H0011,4H0100,
     &4H0101 ,4H0110 ,4H0111 ,4H1000 ,4H1001 ,4H1010 ,4H1011 ,4H1100 ,
     &4H1101 ,4H1110 ,4H1111 /
      DATA (FEBIN(I),I=1,16)/4H0000,4H0001,4H0010,
     &4H0011,4H0100,4H0101,4H0110,4H0111,4H1000,
     &4H1001,4H1010,4H1011,4H1100,4H1101,4H1110,4H1111/
      DATA (FE(I),I=1,16)/4HFE00,4HFE01,4HFE02,4HFE03,
```

```
1050      &4HFE04,4HFE05,4HFE06,4HFE07,4HFE08,4HFE09,
1060      &4HFE10,4HFE11,4HFE12,4HFE13,4HFE14,4HFE15/
1070       DATA (GB(LLL),LLL=1,16)/4HGB00,4HGB01,4HGB02,4HGB03,
1080      &4HGB04,4HGB05,4HGB06,4HGB07,4HGB08,4HGB09,4HGB10,4HGB11,
1090      &4HGB12,4HGB13,4HGB14,4HGB15 /
1100       DATA (RESO(I),I=1,13)/3HPAR,3HRTR,3HCPT,3HTAD,3HTAG,
1110      &3HREX,3HRM1,3HRM2,3HMOD,3HMOG,3HRPO,3HRAD,3HRDE/
1120       DATA (RESOR(I),I=1,13)/4H0000,4H0001,4H0010,
1130      &4H0100,4H0101,4H0111,4H1110,4H1111,4H1100,4H1101,
1140      &4H1010,4H1011,4H0110/
1150       DATA(RU(I),I=1,4)/4HROS ,4HROT ,4HROD ,4HRDS /
1160       DATA (RUR(I),I=1,4)/2H00,2H01,2H10,2H11/
1170       DATA AS(1)/4HIND /
1180       DATA (GBR(LLL),LLL=1,16)/4H0000 ,4H0001 ,4H0010 ,4H0011 ,4H0100 ,
1190      &4H0101 ,4H0110 ,4H0111 ,4H1000 ,4H1001 ,4H1010 ,4H1011 ,4H1100 ,
1200      &4H1101 ,4H1110 ,4H1111 /
1210       DATA (BC(LLL),LLL=1,4)/4HZBC  ,4HBCP  ,4HCPZ  ,4HPTB  /
1220       DATA (BCR(LLL),LLL=1,4)/2H11,2H10,2H01,2H00/
1230       DATA (MO(LLL),LLL=1,2)/4HMAD  ,4HMMD  /
1240       DATA (MOR(LLL),LLL=1,2)/1H1,1H0/
1250       DATA AS1/4H+1  /
1260       DATA(AS1R(LLL),LLL=1,2)/1H0,1H1/
1270       INTEGER REGB2,REGB3,PREVAR,PREVAD
1280       DATA REGB2/1H0/,REGB3/1H1/
1290C      MISE EN FORME DES CHAMPS
1300       DATA SEPAG/1H,SEPAD/1H
1310       DATA BLANC/1H /
1320       INTEGER VBL,VUN,VZE
1330       DATA VBL,VZE,VUN/1H  ,1H0 ,1H1 /
1340       VBL=VBL/2**32
1350       VUN=VUN/2**27
1360       VZE=VZE/2**27
1370       DATA BLANC4/4H    /
1380       INTEGER COM
1390       DATA COM/4H&    /
1400       INTEGER BETQ(7)
1410        INTEGER ETI(7)
1420       INTEGER ERR
1430       NBINST=0
1440       NOMMAC(1)=BLANC
1450       NOMMAC(2)=BLANC
1460       CPTORD=0
1470       INTEGER SUN
1480       DATA SUN/1H1 /
1490       SUN=SUN/2**27*2**27
1500       INTEGER BIN,VAR1,VAR2,NBINS
1510       CALL MIF01(BIN)
1520       CALL MIF02(NBINS,VAR1,VAR2,BIN,VBL,VUN,VZE)
1522       JBIRE=1
1530 5533   CO=0
1540       INDEC=0
1550       TYCH=0
```

```
T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

      NBINST=NBINST+1
       ERR=0
      N=50
      M=50
      J1=0
      J2=0
      N0=0
      J3=0
      DEPL=0
      ERR=0
      NBINST=NBINST+1
      CPTORD=CPTORD+1
      TYSYM=0
       CARSUIV=0
      NODER=0
      READ (5,1234) FIN
      DATA VIRG/1H,/,PTVIRG/1H;/
      PREVAF=0
      PREVAD=0
      NCH=16
      DATA  STAR/4H*    /,QUOTE/1H'/,SEPAR/1H
      READ (5,1) (ETI(I),I=1,7),(ALF(I),I=1,66)
    1 FORMAT(7A1,66A1)
      WRITE(3,1)(ETI(I),I=1,7),(ALF(I),I=1,66)
      ADRUEX=0
    2 FORMAT (6X,66A1)
 1234 FORMAT(I2)
      IF(ETI(1).EQ.STAR) GOTO 4000
      IF(ETI(1).NE.BLANC) GOTO 3000
      IF(ETI(1).EQ.COM ) GOTO 5000
C
C
C
C::::::::::::::::
C;;;SEPARATION DES CHAMPS
C::::::::::::::::
 3001  MICROI=1
      DO 10 J=1,66
      JM=67-J
      IF(ALF(JM).EQ.PTVIRG) GO TO 20
   10 CONTINUE
      CARSUIV=1
      READ(5,2) (ALF(I),I=67,132)
      DO 30 J=67,132
      IF(ALF(J).EQ.PTVIRG) GO TO 40
   30 CONTINUE
      GO TO 5001
   20 J=JM
   40 I1=J-1
      I1=NUMERO CAR AVANT PTVIRG
      IF (ALF(I1).NE.SEPAR ) GO TO 50
      I2=I1-1
```

```
2060    70 IF (ALF(I2).EQ.SEPAG) GO TO 60
2070       I2=I2-1
2080       GO TO 70
2090    60 I3=I2-1
2100C      I3=NUM DERNIER CAR SIGNIF
2110       GO TO 80
2120    50 I3=I1
2130    80 N0=0
2140       K=1
2150    15 IF(ALF(K).EQ.VIRG) GO TO 90
2160       IF(K.EQ.I3) GO TO 05
2170       K=K+1
2180      GOTO 15
2190 90    N0=N0+1
2200       NUM(N0)=K
2210       K=K+1
2220       GO TO 15
2230     5 N0=N0+1
2240       NUM(N0)=K+1
2250       NODER =N0
2260       DO 25 J=1,NODER
2270       L1= NUM(J)-1
2280    65 IF(ALF(L1).EQ.BLANC) GO TO 35
2290       IF(J.EQ.1) GO TO 45
2300       L(J)=L1-NUM(J-1)
2310       GO TO 25
2320    45 L(J)=L1
2330       GO TO 25
2340    35 IF(J.EQ.1) GO TO 55
2350       IF(L1.EQ.NUM(J-1)) GO TO 5004
2360       L1 = L1-1
2370       GO TO 65
2380    55 IF(L1.EQ.0) GO TO 5004
2390       L1=L1-1
2400       GO TO 65
2410    25 CONTINUE
2420    26 CONTINUE
2430C      L(J)=LONG DU CHAMP COMPRIS ENTRE VIRG J-1 ET VIRG J
2440       DO 75 J=1,NODER
2450       IF(ALF(NUM(J)+1).EQ.NUOTE) GOTO 76
2460    75 CONTINUE
2470       IF(J.GT.NODER) GOTO 78
2480    76 JOUOTE=J
2490    78 CONTINUE
2500       DO 77II=1,132
2510       BET(II)=ALF(II)-(ALF(II)/2**27)*2**27
2520    77 ALFA(II)=ALF(II)-BET(II)
2530       DO 85 J=1,NODER
2540       IF(J.EQ.1) GO TO 86
2550       IF(J.EQ.JOUOTE+1) GOTO84
2560       NUMI=NUM(J-1)+1
2570       GO TO 87
```

```
T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

   86 NUMI=1
  87  CONTINUE
      IF(L(J).GT.8) GOTO85
       GOTO (101,102,103,104,105,106,107,108) ,L(J)
T 101 CH(J,1)=ALFA(NUMI)
      GO TO 110
  102 CH(J,1)= ALFA (NUMI)+ALFA(NUMI+1)/2**9
      GO TO 110
  103 CH(J,1)=ALFA(NUMI)+ALFA(NUMI+1)/2**9+ALFA(NUMI+2)/2**18
     &+BLANC/2**27
      GO TO 110
  104 CH(J,1)=ALFA(NUMI)+ALFA(NUMI+1)/2**9+ALFA(NUMI+2)/2**18
      CH(J,1)=CH(J,1)+ALFA(NUMI+3)/2**27
      GO TO 110
  105 CH(J,2) =ALFA(NUMI+4)
      GO TO 115
  106 CH(J,2)=ALFA(NUMI+4)+ALFA(NUMI+5)/2**9
      GO TO 115
  107 CH(J,2)=ALFA(NUMI+4)+ALFA(NUMI+5)/2**9+ALFA(NUMI+6)/2**18
      GOTO 115
  108 CH(J,2)=ALFA(NUMI+4)+ALFA (NUMI+5)/2**9+ALFA(NUMI+6)/2**18
      CH(J,2)=CH(J,2)+ALFA(NUMI+7)/2**27
  115 CH(J,1)=ALFA(NUMI)+ALFA(NUMI+1)/2**9+ALFA(NUMI+2)/2**18
      CH(J,1)=CH(J,1)+ALFA(NUMI+3)/2**27
      GOTO 85
  110 CH(J,2)= BLANC4
  84  CONTINUE
   85 CONTINUE
OC:::::::::::::::
OCETUDE DES CHAMPS
OC SEPARATION DES 4 CAS
OC:::::::::::::::
0     I=1
0     IF((CH(I,1).EQ.DEC).OR.(CH(I,1).EQ.CAL)) GOTO 210
0     IF (CH(I,1).EQ. VAL )  GOTO 220
0     IF (CH(I,1).EQ. ESO)  GOTO 240
0     WRITE (6,61) CAL
0  61 FORMAT (2X,012)
0     GOTO 5007
0C
0C
0C
0C CAS D UNE INSTRUCTION  CAL OU DEC
0  210 IF (CH(I,1).EQ.CAL)GOTO 250
0     VAR(16)= COB(3)
0     CO =2
0     INDEC =1
0     GOTO 305
0  250 VAR(16)= COB(1)
0     CO= 0
0  305 I=I+1
0C:::::::::::::::
```

```
3100CCHAMP PREMIER OPERANDE
3110C:::::::::::::::
3120        DO 310 J=1,16
3130        IF (CH(I,1).EQ.REGEN(J)) GOTO 315
3140   310  CONTINUE
3150        DO 320 J=1,11
3160        IF (CH(I,1).EQ.RCD1(J)) GOTO 325
3170   320  CONTINUE
3180        GOTO 5008
3190 315    J1=J
3200        J3=1
3210        VAR(1)=REGBIN(J)
3220        PREVAR=REGB2
3230        I=I+1
3240        GOTO 330
3250   325  VAR(1) = RCD1B(J)
3260        PREVAR=REGB3
3270        J3=0
3280        I=I+1
3290   330  CONTINUE
3300        IF (INDEC.EQ.1) GOTO 335
3310C::::::::::::::::
3320CCHAMP OPERATEUR
3330C:::::::::::::::
3340C CAS DES INSTRUCTIONS  CAL (CHAMP FC )
3350        DO  340 J =1,4
3360        IF (CH(I,1).EQ.FCC(J))GOTO 345
3370   340  CONTINUE
3380        ERREUR(3)=1
3390        GOTO  5009
3400   345  VAR(2)=  FCBIN (J)
3410        I=I+1
3420        GOTO 350
3430C CAS DES INSTRUCTIONS DEC (CHAMP FC)
3440C:::::::::::::::
3450C CHAMP DESTINATAIRE
3460C:::::::::::::::
3470   335  DO 355 J=1,4
3480        IF (CH(I,1).EQ.FCD(J)) GOTO 360
3490   355  CONTINUE
3500        ERREUR(3)=1
3510        GOTO 5010
3520   360  VAR(2)= FCBIN (J)
3530        I=I+1
3540   350  CONTINUE
3550        DO  365  J=1,16
3560        IF (CH(I,1).EQ.REGEN(J)) GOTO 370
3570   365  CONTINUE
3580        DO 375  J=1,15
3590        IF (CH(I,1).EQ.RCD2(J)) GOTO 380
3600   375  CONTINUE
3610        ERREUR(4)=1
```

```
      GOTO 5011
  370 J2=J
      IF(J3.LE.0) GOTO 385
      IF(J2.EQ.11)GOTO 385
      GOTO 5012
  385 VAR(6)=REGBIN (J)
      PREVAD=REGB2
      I=I+1
      GOTO 395
  380 VAR(6)=RCD2B (J)
      PREVAD=REGB3
      I=I+1
  395 CONTINUE
      GOTO 1000
C
C
C
C CAS D UNE INSTRUCTION SUR VALEUR
  220 VAR(16)= COB(2)
      CO =1
      I=I+1
      DO 405  J=1,16
      IF(CH(I,1).EQ.REGEN(J)) GOTO 410
  405 CONTINUE
      DO 415  J=1,16
      IF (CH(I,1).EQ.RVAL(J)) GOTO 420
  415 CONTINUE
      ERREUR(2)=1
      GOTO 5013
  410 VAR(1)=REGBIN (J)
      PREVAR=REGB2
      I=I+1
      GOTO 425
  420 VAR(1)= RVALB (J)
      PREVAR=REGB3
      I=I+1
  425 CONTINUE
      DO 430 J=1,4
      IF (CH(I,1).EQ.FCC (J)) GOTO 435
  430 CONTINUE
      ERREUR(3)=1
      GOTO 5014
  435 VAR(2)= FCBIN(J)
      I=I+1
  440 CONTINUE
      GOTO 445
C
C
C
C::::::::::::::
C CAS D UNE INSTRUCTION ESO
C::::::::::::::
```

```
4140 240  VAR(16)=COB(4)
4150      CO =3
4160      I=I+1
4170      DO 510 J=1,16
4180      IF (CH(I,1).EQ.REGEN(J)) GOTO 515
4190  510 CONTINUE
4200      DO 520 J=1,14
4210      IF (CH(I,1).EQ.RESO (J)) GOTO 525
4220  520 CONTINUE
4230      ERREUR(2)=1
4240      GOTO 5015
4250  515 VAR(1)=REGBIN(J)
4260      PREVAR=REGB2
4270      I=I+1
4280      GOTO 530
4290  525 VAR(1)= RESOB (J)
4300      PREVAR=REGB3
4310      I=I+1
4320  530 CONTINUE
4330      DO 535 J=1,4
4340      IF(CH(I,1).EQ.FCC(J)) GOTO 540
4350  535 CONTINUE
4360      ERREUR(3)=1
4370      GOTO 5016
4380  540 VAR(2)=FCBIN(J)
4390      I=I+1
4400  545 CONTINUE
4410      DO 550 J=1,16
4420      IF (CH(I,1).EQ.FE(J)) GOTO 555
4430  550 CONTINUE
4440      ERREUR (4)=1
4450      GOTO 5017
4460  555 VAR(6)=FEBIN(J)
4470      I=I+1
4480  560 IF (CH(I,1).EQ.ES(1))GOTO 565
4490      IF(CH(I,1).EQ.ES(2)) GOTO 570
4500      ERREUR(5)=1
4510      GOTO 5018
4520  565 VAR(7) = ESB(1)
4530      GOTO 580
4540  570 VAR(7)=ESB(2)
4550  580 I=I+1
4560  575 CONTINUE
4570      GOTO 1000
4580C
4590C
4600C
4610C::::::::::::::
4620C  ETUDE DU CHAMP VALEUR
4630C::::::::::::::::
4640 445  LV=ALE(NUM(3)+1)
4650      NO=NUM(3)+1
```

```
T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

      IF (LV.NE.QUOTE) GOTO 605
      IF (CH(2,1).EQ.RCD2(4))GOTO 5050
      N1=N0+1
 615  IF((ALF(N1).NE.UN).OR.(ALF(N1).NE.ZERO)) GOTO610
      N1=N1+1
      GOTO 615
 610  IF(N1-N0.EQ.9) GOTO 620
      GOTO 5019
  620 IF (ALF(N1).NE.QUOTE) GOTO 5020
C ECRITURE DU CHAMP
 631  VAR(6)=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18+
     &ALFA(N0+4)/2**27
      VAR2(6)=ALFA(N0+5)+ALFA(N0+6)/2**9+ALFA(N0+7)/2**18
     &+ALFA(N0+8)/2**27
      TYCH    = 2
C TYCH=2,VAL EN BINAIRE
      GOTO  1100
 605  CONTINUE
      IF((LV.NE.PLUS).AND.(LV.NE.MOINS)) GOTO625
      DEPL=1
      N1=N0+1
      GOTO 630
 637  N0=N0-1
      N1=N0+1
 630  IF((ALF(N1).LT.CHI(10)).OR.(ALF(N1).GT.CHI(9))) GOTO 638
      ALFI(N1)=(ALFA(N1)-ALFA0)/2**27
      N1=N1+1
      GOTO 630
  638 IF(N1-N0.GT.4) GOTO 5022
      GOTO (621,622,623),N1-N0-1
 621  VALD=ALFA(N0+1)
      VALDEC=ALFI(N0+1)
      GOTO 640
 622  VALD=ALFA(N0+1)+ALFA(N0+2)/2**9
      VALDEC=ALFI(N0+1)*10+ALFI(N0+2)
      GOTO 640
 623  VALD=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+1)/2**18
      VALDEC=ALFI(N0+1)*100+ALFI(N0+2)*10+ALFI(N0+3)
      MARQ=1
 640  CONTINUE
      IF(DEPL.NE.1) GOTO 645
      IF(VALDEC.GT.256) GOTO 5023
      IF(ALF(N0).EQ.PLUS) GOTO 650
      VALDEC=-VALDEC
 650  VALDEC=VALDEC+CPTORD
      TYCH=3
      VAR2(6)=BLANC4
     CALL DECBIN(VALDEC,VAR2(6),VAR(6),VBL,VZE,VUN )
C
C TYCH=3   VAL=DEPLACEMENT
      GOTO 1100
 645  VAR(6)=VALDEC
```

```
5180        VAR2(6)=BLANC4
5190        TYCH=1
5200        CALL DECBIN(VALDEC,VAR2(6),VAR(6),VBL,VZE,VUN)
5210C
5220C TYCH=1   VAL=NOMBRE DECIMAL
5230        GOTO 1100
5240 625    CONTINUE
5250        ALFES =ALF(N0)
5260        IF((ALF(N0).LT.CHI(10)).OR.(ALF(N0).GT.CHI(9))) GOTO 655
5270        DEPL=0
5280        GOTO 637
5290 655    CONTINUE
5300        IF((ALF(N0).LT.ALPH(1)).OR.(ALF(N0).GT.ALPH(26))) GOTO 5025
5310        N1=N0+1
5320 670    CONTINUE
5330        IF((ALF(N1).EQ.BLANC).OR.(ALF(N1).EQ.VIRG)) GOTO 665
5340        IF(((ALF(N1).GE.ALPH(1)).AND.(ALF(N1).LE.ALPH(26))).OR.
5350       &((ALF(N1).GE.CHI(10)).AND.(ALF(N1).LE.CHI(9)))) GOTO 660
5360        GOTO 665
5370 660    N1=N1+1
5380        GOTO 670
5390 665     IF(N1-N0.GE.6) GOTO5026
5400        N0=N0-1
5410        GOTO (671,672,673,674,675,676),N1-N0-1
5420 671     SYMB1=ALFA(N0+1)
5430         GOTO 677
5440 672     SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9
5450         GOTO 677
5460 673    SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
5470        GOTO 677
5480 674    SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
5490       &+ALFA(N0+4)/2**27
5500 677    SYMB2=BLANC4
5510        VAR(6)=SYMB1
5520        VAR2(6)=SYMB2
5530        GOTO 680
5540 675    SYMB2=ALFA(N0+5)
5550        GOTO 678
5560  676  SYMB2=ALFA(N0+5)+ALFA(N0+6)/2**9
5570 678    SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
5580       &+ALFA(N0+4)/2**27
5590        CALL  SCHVAL (SYMB1,SYMB2,NPROG1,NPROG2,NBINST,NBPARA
5600       &,TABSYM,PLASYM,N,M,ERR)
5610        VAR(6)=SYMB1
5620        VAR2(6)=SYMB2
5630 680     CONTINUE
5640        VAR(6)=SYMB1
5650        VAR2(6)=SYMB2
5660        TYCH=4
5670        GOTO 685
5680C
5690C TYCH=4   VAL=SYMBOLE
```

```
T 01   08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

 684   N1=N1+1
 685    CONTINUE
       IF(ALF(N1).EQ.BLANC) GOTO 684
       IF((ALF(N1).EQ.PLUS).OR.(ALF(N1).EQ.MOINS)) GOTO 690
       IF((ALF(N1).EQ.VIRG).OR.(ALF(N1).EQ.PTVIRG)) GOTO 1100
       GOTO 5071
 690   N0=N1
       N1=N0+1
       DEPL2=0
       IF(ALF(N0).EQ.MOINS) GOTO 691
       GOTO 692
 691   DEPL2=1
 692   CONTINUE
 700   IF((ALF(N1).LT.CHI(10)).OR.(ALF(N1).GT.CHI(9))) GOTO695
       ALFI(N1)=(ALF(N1)-ALFA0)/2**27
       N1=N1+1
        GOTO 700
  695  IF(N1-N0.GT.4) GOTO 5028
       GOTO (701,702,703),N1-N0-1
 701   VALD=ALFA(N0+1)
       VALDEC=ALFI(N0+1)
         GOTO 705
 702   VALD=ALFA(N0+1)+ALFA(N0+2)/2**9
       VALDEC=ALFI(N0+1)*10+ALFI(N0+2)
       GOTO 705
 703   VALD=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
       VALDEC=ALFI(N0+1)*100+ALFI(N0+2)*10+ALFI(N0+3)
  705  IF(VALDEC.GT.256) GOTO 5029
       IF(DEPL2.EQ.1) GOTO 706
       GOTO 708
 706   VALDEC=-VALDEC
 708   CONTINUE
       TYCH=5
       VAR(6)=SYMB1
       VAR2(6)=VALDEC
00
00 TYCH=5   VAL=SYMBOLE+DEPLACEMENT
       GOTO 1100
1100   I=I+1
       DO 900 J = 1,4
       IF(CH(I,1).EQ.RU(J)) GOTO905
  900  CONTINUE
       GOTO 5050
  905  VAR(5)=RUB (J)
       I=I+1
       GOTO  1000
1000   CONTINUE
00::::::::::::
00 ETUDE DES CHAMPS FACULTATIF
00::::::::::::
       IF(I.GT.NODEF) GOTO2010
       IF((CC.EQ.1).OR.(CC.EQ.3)) GOTO 1010
```

```
6220        DO 1015  J=1,4
6230        IF (CH(I,1).EQ.DO(J)) GOTO 1020
6240 1015 CONTINUE
6250        VAR(9)=DOB(1)
6260        GOTO 1025
6270 1020 VAR(9)=DOB(J)
6280        I=I+1
6290 1025 IF(I.GT.NODER) GOTO 2020
6300        GOTO 1030
6310 1010 IF (CO.EQ.1) GOTO 1035
6320 1030 DO 1040 J =1,4
6330        IF (CH(I,1).EQ.RI(J)) GOTO 1045
6340 1040 CONTINUE
6350        VAR(3) =RIB(1)
6360        GOTO 1050
6370 1045 IF((J.EQ.4).AND.(INDEC.EQ.0)) GOTO 5051
6380        VAR(3) =RIB(J)
6390 1046 I=I+1
6400 1050 IF (I.GT.NODER) GOTO 2030
6410 1035 DO 1055 J =1,2
6420        IF (CH(I,1).EQ.RP(J)) GOTO 1060
6430 1055 CONTINUE
6440        VAR(4) =RPB(2)
6450        GOTO 1065
6460 1060 VAR(4) =RPB (J)
6470        I=I+1
6480 1065 IF (I.GT.NODER) GOTO 2040
6490        DO 1070  J=1,3
6500        IF (CH(I,1).EQ.PT(J)) GOTO 1075
6510 1070 CONTINUE
6520        VAR (5)= PTB (4)
6530        GOTO 1080
6540 1075 VAR(5) =PTB (J)
6550        I=I+1
6560 1080 IF (I.GT.NODER) GOTO 2050
6570        IF (CO.EQ.1) GOTO  1085
6580        IF (CH(I,1).EQ.FT)  GOTO 1090
6590        VAR (8) =FTB(1)
6600        IF(CO.EQ.3) VAR2(8)=FTB(1)
6610        GOTO 1095
6620 1090 VAR (8) =FTB (2)
6630        IF(CO.EQ.3) VAR2(8)=FTB(2)
6640        I=I+1
6650 1095 IF(I.GT.NODER) GOTO 2060
6660 1085 DO 1101 J = 1,2
6670        IF (CH(I,1).EQ.SP(J)) GOTO 1105
6680 1101 CONTINUE
6690        VAR(14) = SPB(2)
6700        GOTO 1110
6710 1105 VAR(14) =SPB(J)
6720        I=I+1
6730 1110 IF (I.GT.NODER) GOTO 2070
```

```
T 01   08-20-75   11.697      CONVERSION DE SUPPORT        PAR BFRINT

       IF (CO.NE.3) GOTO 1115
       DO 1120 J=1,2
       IF (CH(I,1).EQ. RV(J)) GOTO 1125
 1120  CONTINUE
       VAR(9) = RVP(2)
       GOTO 1130
 1125  VAR (9) =RVB (J)
       I=I+1
 1130  IF (I.GT.NODER) GOTO  2080
       IF(CH(I,1).EQ.AS(1)) GOTO1135
       VAR(10) = ASB (1)
       GOTO  1140
 1135  VAR(10)=ASB (2)
       I=I+1
 1140  IF(I.GT.NODER) GOTO 2090
       DO 1145 J=1,16
       IF (CH(I,1).EQ. AE(J)) GOTO  1150
 1145  CONTINUE
       VAR(11)=AEB(1)
       GOTO  5052
 1150  VAR(11) =AEB (J)
 1151  I=I+1
       IF(I.GT.NODER) GOTO 2120
       GOTO 1155
 1115  CONTINUE
       DO 1160 J=1,16
       IF (CH(I,1).EQ.GB(J)) GOTO 1165
 1160  CONTINUE
       GOTO  1200
 1165  VAR(11)=GBB(J)
       VAR(15)=AS1B(1)
       I=I+1
       IF (I.GT.NODER) GOTO  2100
 1155  DO 1170 J=1,4
       IF (CH(I,1).EQ. BC(J)) GOTO 1175
 1170  CONTINUE
       VAR (12) = BCB(4)
       GOTO 1180
 1175  VAR (12) =BCB(J)
       I=I+1
 1180  IF( I.GT.NODER)  GOTO 2110
       DO 1185 J = 1,2
       IF (CH(I,1).EQ. MO(J)) GOTO 1190
 1185  CONTINUE
       VAR(13) =MOB (2)
       GOTO  1195
 1190  VAR(13) =MOB (J)
       I=I+1
 1195  IF (CO.NE.3) GOTO 1211
       IF (CH(I,1).EQ.AS1) GOTO 1206
 1211  VAR(15)= AS1B(1)
       GOTO 1300
```

```
7260 1206 VAR(15)=AS1B(2)
7270 1300 CONTINUE
7280      GOTO 2111
7290C  TOUS LES CHAMPS SONT TRADUITS EN BIN
7300C::::::::::::
7310C CAS D'UNE ADRESSE DE RUPTURE
7320C CHAMP ADR DE RUPTURE
7330C::::::::::::
7340 1200 N0=NUM(I-1)+1
7350      ADRUEX=1
7360      LV=ALF (N0)
7370      IF(LV.NE.QUOTE) GOTO 1205
7380      N1=N0+1
7390 1215  IF((ALF(N1).NE.UN).OR.(ALF(N1).NE.ZERO)) GOTO 1210
7400      N1=N1+1
7410        GOTO 1215
7420 1210 IF(N1-N0.EQ.9) GOTO 1220
7430      GOTO 5060
7440 1220 IF(ALF(N1).NE.QUOTE) GOTO 5061
7450      VAR(11)=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
7460     &+ALFA(N0+4)/2**27
7470      VAR2(11)=ALFA(N0+5)+ALFA(N0+6)/2**9+ALFA(N0+7)/2**18
7480     &+ALFA(N0+8)/2**27
7490      TYCH=7
7500      VAR(15)=AS1B(2)
7510      GOTO 1300
7520 1205 IF((LV.NE.PLUS).AND.(LV.NE.MOINS)) GOTO 1225
7530       DEPL=1
7540      N1=N0+1
7550      GOTO 1230
7560 1237 N0=N0-1
7570      N1=N0+1
7580 1230 IF((ALF(N1).LT.CHI(10)).OR.(ALF(N1).GT.CHI(9))) GOTO 1238
7590      ALFI(N1)=(ALFA(N1)-ALFA0)/2**27
7600       N1=N1+1
7610       GOTO 1230
7620 1238 IF(N1-N0.GT.4) GOTO 5069
7630      GOTO (1231,1232,1233),N1-N0-1
7640 1231 VALD=ALFA(N0+1)
7650      VALDEC=ALFI(N0+1)
7660      GOTO 1240
7670 1232 VALD=ALFA(N0+1)+ALFA(N0+2)/2**9
7680      VALDEC=ALFI(N0+1)*10+ALFI(N0+2)
7690      GOTO 1240
7700 1233 VALD=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
7710      VALDEC=ALFI(N0+1)*100+ALFI(N0+2)*10+ALFI(N0+3)
7720 1240 IF(DEPL.NE.1) GOTO1245
7730      IF(VALDEC.GT.256) GOTO 5070
7740       IF(ALF(N0).EQ.PLUS) GOTO 1250
7750      VALDEC=-VALDEC
7760 1250 VALDEC=VALDEC+CPTORD
7770      VAR2(11)=BLANC4
```

```
'T 07   08-20-75   11.697      CONVERSION DE SUPPORT        PAR DFRINT

)       TYCH=8
)       CALL DECBIN(VALDEC,VAR2(11),VAR(11),VBL,VZE,VUN)
)       VAR(15)=AS1B(2)
)       GOTO 1300
) 1245  VAR(11)=VALDEC
)       VAR2(11)=BLANC4
)       CALL DECBIN(VALDEC,VAR2(11),VAR(11),VBL,VZE,VUN)
)       TYCH=6
)       VAR(15)=AS1B(2)
)       GOTO 1300
) 1225  CONTINUE
)       IF((ALF(N0).LT.CHI(10)).OR.(ALF(N0).GT.CHI(9))) GOTO 1255
)       DEPL=0
)       GOTO 1257
) 1255  CONTINUE
)       IF((ALF(N0).LT.ALPH(1)).OR.(ALF(N0).GT.ALPH(26))) GOTO 5066
)       N1=N0+1
) 1270  IF((ALF(N1).EQ.BLANC).OR.(ALF(N1).EQ.VIRG)) GOTO 1265
)       IF(((ALF(N1).GE.ALPH(1)).AND.(ALF(N1).LE.ALPH(26))).OR.
)      &((ALF(N1).GE.CHI(10)).AND.(ALF(N1).LE.CHI(9)))) GOTO 1260
)       GOTO 1265
) 1260  N1=N1+1
0       GOTO 1270
0 1265  IF(N1-N0.GT.6) GOTO 5067
0       N0=N0-1
0       GOTO (1271,1272,1273,1274,1275,1276),N1-N0-1
0 1271  SYMB1=ALFA(N0+1)
0       GOTO 1277
0 1272  SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9
0        GOTO 1277
0 1273  SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
0       GOTO 1277
0 1274  SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
0      &+ALFA(N0+4)/2**27
0 1277  SYMB2=BLANC4
0       GOTO 1280
0 1275  SYMB2=ALFA(N0+5)
0       GOTO 1278
0 1276  SYMB2=ALFA(N0+5)+ALFA(N0+6)/2**9
0 1278  SYMB1=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
0      &+ALFA(N0+4)/2**27
0 1280   CONTINUE
0       VAR(11)=SYMB1
0       VAR2(11)=SYMB2
0       VAR(11)=SYMB1
0       VAR2(11)=SYMB2
0       TYCH=9
0       GOTO 1285
0 1284  N1=N1+1
0 1285  CONTINUE
0        IF(ALF(N1).EQ.BLANC) GOTO 1284
0       IF((ALF(N1).EQ.PLUS).OR.(ALF(N1).EQ.MOINS)) GOTO 1290
```

```
0267T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BERINT

8300        IF((ALF(N1).EQ.VIRG).OR.(ALF(N1).EQ.PTVIRG)) GOTO 1300
8310        GOTO5071
8320 1290    N0=N1
8330        N1=N0+1
8340        DEPLS=0
8350        IF(ALF(N0).EQ.MOINS) GOTO 1291
8360        GOTO 1292
8370 1291 DEPLS=1
8380 1292 CONTINUE
8390 1400 IF((ALF(N1).LT.CHI(10)).OR.(ALF(N1).GT.CHI(9))) GOTO1295
8400        ALFI(N1)=(ALF(N1)-ALFA0)/2**27
8410        N1=N1+1
8420        GOTO 1400
8430 1295 IF(N1-N0.GT.4) GOTO 5063
8440        GOTO  (1401,1402,1403),N1-N0-1
8450 1402 VALD=ALFA(N0+1)+ALFA(N0+2)/2**9
8460        VALDEC=ALFI(N0+1)*10+ALFI(N0+2)
8470        GOTO 1405
8480 1401 VALD=ALFA(N0+1)
8490        VALDEC=ALFI(N0+1)
8500        GOTO 1405
8510 1403 VALD=ALFA(N0+1)+ALFA(N0+2)/2**9+ALFA(N0+3)/2**18
8520        VALDEC=ALFI(N0+1)*100+ALFI(N0+2)*10+ALFI(N0+3)
8530 1405 IF(VALDEC.GT.256) GOTO 5064
8540        IF(DEPLS.EQ.1) GOTO 1406
8550        GOTO 1408
8560 1406 VALDEC=-VALDEC
8570 1408 CONTINUE
8580        TYCH=10
8590        VAR(11)=SYMB1
8600        VAR2(11)=VALDEC
8610        VAR(15)=AS1B(2)
8620        GOTO 1300
8630C ::::::::::::
8640C COMMENTAIRES CONCERNANT LES ERREURS
8650C::::::::::::
8660 5001 WRITE (6,6001)
8670 6001 FORMAT ('INSTRUCTION INCOMPLETE -PTVIRG MANQUANT RAJOUTE.')
8680        J=131
8690 7002 IF((ALF(J).EQ.BLANC).OR.(ALF(J).EQ.VIRG).OR.
8700       &(ALF(J).EQ.SEPAG)) GOTO7001
8710        J=J-1
8720        GOTO 7002
8730 7001 GOTO 20
8740 5004 WRITE (6,6004)  K
8750 6004 FORMAT(8HLE CHAMP     ,I2,28H NE CONTIENT QUE DES BLANCS      )
8760       ERR=1
8770        GOTO 26
8780 5007 WRITE(6,6007)
8790 6007 FORMAT ('***  LE CHAMP CO EST FAUX -INSTRUCTION A REPRENDRE ***')
8800        ERR=1
8810        GOTO 2222
```

```
T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR EPRINT
 5008 WRITE(6,6008)
 6008 FORMAT (' ***   CHAMP NR FAUX : REGISTRE NON AUTORISE ***')
      ERR=1
      GOTO  350
 5009 WRITE(6,6009)
 6009 FORMAT ('***  CHAMP FC FAUX :  CONTENU NON AUTORISE *** ')
      ERR=1
      GOTO  350
 5010 WRITE(6,6010)
 6010 FORMAT ('***  CHAMP FC FAUX :  CONTENU NON AUTORISE *** ')
      ERR=1
      GOTO 350
 5011 WRITE(6,6011)
 6011 FORMAT  ('***  CHAMP  NR2 FAUX : REGISTRE NON AUTORISE ***')
      ERR=1
      GOTO 395
 5012 WRITE(6,6012)
 6012 FORMAT (50H*** 2 REG GENERAUX DIFFERENTS SONT UTILISES COMME'
     &38H PREMIER OPERANDE ET DESTINATAIRE ***      )
      ERR=1
      GOTO 395
 5013 WRITE(6,6013)
 6013 FORMAT ('***  CHAMP NR FAUX : REGISTRE NON AUTORISE *** ' )
      ERR=1
      GOTO 425
 5014 WRITE(6,6014)
 6014 FORMAT ('*** CHAMP FC FAUX : CONTENU NON AUTORISE *** ')
      ERR=1
      GOTO 440
 5015 WRITE(6,6015)
 6015 FORMAT ('*** CHAMP NR FAUX : REGISTRE NON AUTORISE ***')
     ERR=1
      GOTO  530
 5016 WRITE(6,6016)
 6016 FORMAT ('*** CHAMP FC FAUX : CONTENU NON AUTORISE  ***')
      ERR=1
      GOTO  545
 5017 WRITE(6,6017)
 6017 FORMAT ('*** CHAMP FE FAUX :CONTENU NON AUTORISE *** ')
 ERR =1
      GOTO 560
 5018 WRITE(6,6018)
 6018 FORMAT('*** CHAMP ES FAUX: CONTENU NON AUTORISE *** ')
 ERR=1
      GOTO 575
 5019 WRITE(6,6019)
 6019 FORMAT ('*** CHAMP VALEUR :ECRITURE BINAIRE TROP LONGUE ***')
 ERR=1
      GOTO  620
 5020 WRITE(6,6020)
 6020 FORMAT('*** CHAMP VALEUR: ABSCENCE  QUOTE FINALE ***')
 ERR=1
```

```
9340        GOTO 631
9350 5022 WRITE(6,6022)
9360 6022 FORMAT(45H***CHAMP VALEUR : VAL DECIM TROP GRANDE          )
9370 ERR=1
9380         GOTO 1100
9390 5023 WRITE(6,6023)
9400 6023 FORMAT(40H*** CHAMP VALEUR: DEPL TROP GRAND          )
9410 ERR=1
9420        GOTO 1100
9430 5025 WRITE(6,6025)
9440 6025 FORMAT(55H*** CHAMP VALEUR : PREMIER CARACTERE SYMBOLE NON LETTRE
9450      &4H ***  )
9460        ERR=1
9470        GOTO  1100
9480 5026 WRITE(6,6026)
9490 6026 FORMAT ('*** CHAMP VALEUR : LONGUEUR SYMBOLE NON AUTORISEE ***')
9500 ERR=1
9510        GOTO  1100
9520 5028 WRITE(6,6028)
9530 6028 FORMAT ('*** CHAMP VALEUR : VALEUR  DECIMALE TROP IMPORTANTE ***')
9540 ERR=1
9550        GOTO  1100
9560 5029 WRITE(6,6029)
9570 6029 FORMAT ('*** CHAMP VALEUR : VALEUR DECIMALE TROP GRANDE ***')
9580 ERR=1
9590        GOTO   1100
9600 5030 WRITE(6,6030)
9610 6030 FORMAT (51H*** CHAMP VALEUR : REGISTRE TAG UTILISE AVEC VALEUR',
9620      &14H BINAIRE *** ' )
9630        ERR=1
9640        GOTO  1100
9650 5063 WRITE(6,6063)
9660 6063 FORMAT ('*** CHAMP SU :  VALEUR DEPLACEMENT TROP DE CHIFFRE ***')
9670 ERR=1
9680        GOTO 2222
9690 5064 WRITE(6,6064)
9700 6064 FORMAT( 50H*** CHAMP SU : VALEUR DEPLACEMENT TROP GRANDE ***' )
9710 ERR=1
9720        GOTO 2222
9730 5066 WRITE(6,6066)
9740 6066 FORMAT ('*** CHAMP SU : PREMIER CARACTERE SYMBOLE NON LETTRE ***'
9750 ERR=1
9760        GOTO 2222
9770 5067 WRITE (6067)
9780 6067 FORMAT(40H*** CHAMP SU:LONG SYMB INTERDITE          )
9790 ERR=1
9800        GOTO 2111
9810 5050 WRITE(6,6050 )
9820 6050 FORMAT ('*** ABSCENCE DU CHAMP RU *** ')
9830 ERR=1
9840        GOTO 1000
9850 5051 WRITE(6,6051)
```

```
'T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

) 6051 FORMAT (54H***CHAMP R4 : RIB UTILISE DANS UNE INSTRUCTION AUTRE ',
)      &18HQUE DECALAGE *** ')
) ERR=1
)       GOTO 1046
) 5052 WRITE(6,6052)
) ERR=1
) 6052 FORMAT (45H*** CHAMP AE D'UNE INSTRUCTION ESO VIDE  ***')
)       GOTO 1151
) 5060 WRITE(6,6060)
) 6060 FORMAT (57H*** CHAMP   SO      : LONGUEUR DU BINAIRE DIFFERENTE DE ',
)      &6H8 *** )
) ERR=1
)       GOTO 1300
) 5061 WRITE(6,6061)
00 6061 FORMAT (51H*** CHAMP  SO : ABSCENCE DE QUOTE EN FIN DU CHAMP ',
10      &12HBINAIRE ***')
20 ERR=1
30       GOTO 1300
40 5069 WRITE(6,6069)
50 6069 FORMAT (54H*** CHAMP SO : NOMBRE DE CHIFFRE DECIMAUX TROP IMPOR',
60      &10HTANT ***  )
70 ERR=1
80       GOTO 1300
90 5070 WRITE(6,6070)
00 6070 FORMAT ('*** CHAMP SO : VALEUR DECIMALE TROP GRANDE ***')
10 ERR=1
20       GOTO 1300
30 5071  WRITE(6,6071)
40 6071 FORMAT('*** ERREUR : CARACTERE INCORRECT APRES UN SYMB :: ')
50 ERR=1
60        GOTO 1300
70C::::::::::
80C ETUDE DES CHAMPS FACULTATIFS PRIS PAR DEFAUT
90C::::::::::::
00C CHAMP PRIS PAR DEFAUT  APRES BC
10 2010 IF(CO.EQ.1) GOTO 2030
20        IF(CO.EQ.3) GOTO 2020
30        VAR(9)=DOB(1)
40 2020 VAR(3)=RIB(1)
50 2030 VAR(4)=RPB(2)
60 2040 VAR(5)=PTB(4)
70 2050 IF(CO.EQ.1) GOTO 2060
80       VAR(8)=FTB(1)
90       IF(CO.EQ.3) VAR2(8)=FTB(1)
00 2060 VAR(14)=SPB(2)
10       GOTO 2070
20 2075 VAR(11)=GBB(1)
30 2100 VAR(12)=BCB(4)
40       GOTO 2110
50 2070 IF(CO.NE.3) GOTO 2075
60        VAR(9)=RVB(2)
70 2080 VAR(10)=ASB(1)
```

```
0267T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

10380 2090 VAR(11)=AEB(1)
10390 2120 VAR(12)=BCB(4)
10400 2110 VAR(13)=MOB(2)
10410      VAR(15)=AS1B(1)
10420 2111 IF(ADRUEX.EQ.1) GOTO 2112
10430C
10440C CAS PAS D'ADRESSE DE RUPTURE
10450C
10460      IF((CO.EQ.1).OR.(CO.EQ.3)) GOTO 2250
10470C
10480C CAS INSTRUCTION DE CALCUL OU DE DECALAGE
10490      WRITE (6,2510) PREVAR,(VAR(I),I=1,5),PREVAD,VAR(6),
10500     &VAR(8),VAR(9),VAR2(8),
10510     &(VAR(I),I=11,16)
10520 2510 FORMAT(5X,A1,1X,A4,1X,2(A2,1X),A1,1X,A2,1X,A1,1X,A4,1X,
10530     &1X,A1,1X,A2,1X,A1,1X,A4,1X,A2,3(1X,A1),1X,A2)
10540      GOTO 2222
10550 2250 IF(CO.EQ.1) GOTO 2260
10560C
10570C CAS INSTRUCTION ENTREE-SORTIE
10580      WRITE(6,2520) PREVAR,(VAR(I),I=1,10),VAR2(8),(VAR(I),I=11,16)
10590 2520 FORMAT(5X,A1,1X,A4,2(1X,A2),1X,A1,1X,A2,1X,A4,5(1X,A1),1X,A4,
10600     &1X,A2,3(1X,A1),1X,A2)
10610      GOTO 2222
10620C
10630C CAS INSTRUCTION SUR VALEUR
10640 2260 GOTO (2261,2262,2261,2262,2265),TYCH
10650 2261 WRITE (6,2531) PREVAR,(VAR(I),I=1,6),VAR2(6),
10660     &(VAR(I),I=11,16)
10670 2531 FORMAT(5X,A1,1X,A4,2(1X,A2),1X,A1,1X,A2,1X,A4,A4,1H*,
10680     &A4,1X,A2,3(1X,A1),1X,A2)
10690      GOTO 2222
10700 2262 WRITE(6,2532) PREVAR,(VAR(I),I=1,5),VAR(6),VAR2(6),
10710     &(VAR(I),I=11,16)
10720 2532 FORMAT(5X,1H*,A1,1H*,A4,2(1H*,A2),1H*,A1,1H*,A2,1H*,2A4,2H* ,
10730     &1H*,A4,1H*,A2,3(1H*,A1),1H*,A2)
10740      GOTO 2222
10750 2265 WRITE (6,2535) PREVAR,(VAR(I),I=1,6),VAR2(6),(VAR(I),I=11,16)
10760 2535 FORMAT(5X,1H*,A1,1H*,A4,2(1H*,A2),1H*,A1,1H*,A2,1H*,A4,I4,
10770     &2H* ,1H*,A4,1H*,A2,3(1H*,A1),1H*,A2)
10780      GOTO 2222
10790 2112 IF((CO.EQ.1).OR.(CO.EQ.3)) GOTO2270
10800      WRITE(6,2540) PREVAR,(VAR(I),I=1,5),PREVAD,
10810     &VAR(6),VAR(8),VAR(9),VAR(11),VAR2(11),(VAR(I),I=14,16)
10820 2540 FORMAT(5X,A1,1X,A4,2(1X,A2),1X,A1,1X,A2,
10830     &1X,A1,1X,A4,1X,A1,1X,A2,2(1X,A4),2(1X,A1),1X,A2)
10840      GOTO 2222
10850 2270 IF(CO.EQ.1) GOTO 2280
10860      GOTO 2222
10870 2280  WRITE (6,2550) PREVAR,(VAR(I),I=1,5),VAR(6),VAR2(6),
10880     &VAR(11),VAR2(11),(VAR(I),I=14,16)
10890
```

```
T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

) 2550 FORMAT(5X,1H:,A1,1H:,A4,1H:,2(1H:,A2),1H:,A1,1H:,A2,1H:,
)     &A4,A4,1H:,2A4,1H:,2(A1,1H:),A2)
)      GOTO 2222
) 4000 CONTINUE
)C ::::::::::::::::::::::::
)C SELECTION DE LA PSEUDO UTILISEE
)C::::::::::::::::::::
)      N=50
)      M=50
)      DO 7101 I=2,7
)            BETQ(I)=ETI(I)-ETI(I)/2**27*2**27
) 7101 ETI(I)=ETI(I)-BETQ(I)
)      DO 7010 I=1,66
)      BET(I)=ALF(I)-(ALF(I)/2**27)*2**27
) 7010 ALFA(I)=ALF(I)-BET(I)
)      TYPE=ALFA(1)+ALFA(2)/2**9+ALFA(3)/2**18+ALFA(4)/2**27
)      DO 7020 J=1,8
)      IF(TYPE.EQ.TYPO(J)) GOTO 7030
) 7020 CONTINUE
)      IF(TYPE.NE.TYPO(8))  GOTO 5150
) 7030 IF(J.GE.7) GOTO 7040
)      SYMB1=ALFA(6)+ALFA(7)/2**9+ALFA(8)/2**18+ALFA(9)/2**27
)      SYMB2=ALFA(10)+ALFA(11)/2**9
)      GOTO 7005
) 7040 IF(J.EQ.8) GOTO7005
)      SYMB1=ETI(2)+ETI(3)/2**9+ETI(4)/2**18+ETI(5)/2**27
)      SYMB2=ETI(6)+ETI(7)/2**9
) 7005 GOTO (7011,7012,7013,7014,7015,7016,7017,7018) ,J
) 7011 CALL CRPROG (SYMB1,SYMB2,TABSYM,NBPARA,TAIPRO,IPRO,
)     &NBINST,NINSTM,NIVMAC,N,M,ERR)
)      NPROG1=SYMB1
)      NPROG2=SYMB2
)      WRITE (6,6021) NPROG1,NPROG2
) 6021 FORMAT('NPROG1=',A4,2X,'NPROG2=',A4)
)      DO 1122 NG=1,8
) 6027 FORMAT('U6027'/,'TABSYM',/,6(2X,O12))
)      WRITE (6,6027 )(TABSYM(NG,NH),NH=1,6)
) 1122 CONTINUE
)      GOTO 7100
) 7012 IF(ALF(2).EQ.BLANC) GOTO 7025
)      ETIQ1=ETI(2)+ETI(3)/2**9+ETI(4)/2**18+ETI(5)/2**27
)      ETIQ2=ETI(6)+ETI(7)/2**9
) 7025 CALL FINPROG(SYMB1,SYMB2,ETIQ1,ETIQ2,TAIPRO,CPTORD,NBINST,N,
)     &ERR,IPRO )
)      GOTO 7100
) 7013 CALL PDEF (SYMB1,SYMB2,NPROG1,NPROG2,NBPARA,TABSYM,N,M,ERR)
)      GOTO 7100
) 7014 CALL PREF (SYMB1,SYMB2,NPROG1,NPROG2,NBPARA,TABSYM,N,ERR)
)      GOTO 7100
) 7015 CALL CRMACR (SYMB1,SYMB2,TABSYM,NBPARA,TMACRP,IMACRP,NINSTM,
)     &CPTORD,NIVMAC,TRMACRP,NOMMAC,N,M)
)      GOTO 7100
```

```
11420 7016 CALL REFMAC (SYMB1,SYMB2,NOMMAC,TMACRP,IMACRP,PMACRP,FMACRP
11430     &,TMACRO,IMACRO,PMACRO,FMACRO,BINREL,NBINST,NBINSTP,NIVMAC,
11440     &CPTORD,CPTORM,N,M,L,LP,LO)
11450      IF(ALF(2).EQ.BLANC ) GOTO 7035
11460      ETIQ1=ALF(2)+ALF(3)/2**9+ALF(4)/2**18+ALF(5)/2**27
11470      ETIQ2=ALF(6)+ALF(7)/2**9
11480      CALL CRSYMB (ETIQ1,ETIQ2,NPROG1,NPROG2,NOMMAC,CPTORD,NBINST,PLASY
11490     &NBPARA,ERR,TABSYM,N,M)
11500      GOTO 7100
11510 7017 I=6
11520 7048  IF((ALF(I).EQ.BLANC).OR.(ALF(I).EQ.VIRG)) GOTO 7035
11530      I=I+1
11540      GOTO 7048
11550 7035 GOTO (7041,7042,7043,7044),I-6
11560 7041 VALEUR=ALFA(6)
11570      GOTO 7050
11580 7042 VALEUR=ALFA(6)+ALFA(7)/2**9
11590      GOTO 7050
11600 7043 VALEUR=ALFA(6)+ALFA(7)/2**9+ALFA(8)/2**18
11610      GOTO 7050
11620 7044 VALEUR=ALFA(6)+ALFA(7)/2**9+ALFA(8)/2**18+ALFA(9)/2**27
11630      GOTO 7050
11640
11650 7050 CALL CREGAL  (SYMB1,SYMB2,NPROG1,NPROG2,VALEUR,NBPARA,TABSYM,N,
11660     &ERR)
11670      GOTO 7100
11680 7018 CALL CRPEND (TMACRP,IMACRP,CPTORM,NOMMAC,NIVMAC,M)
11690 7100 IF(ERR.EQ.1) GOTO 7080
11700      GOTO 2222
11710 7080 WRITE (6,5151)
11720 5151 FORMAT (40H*** ERREUR :PSEUDO NON AUTORISEE                    )
11730      GOTO 2222
11740 5150 WRITE (6,6150)
11750 6150 FORMAT(30H***ERREUR PSEUDO INCONNUE **      )
11760      GOTO 2222
11770 3000 CONTINUE
11780      DO 7103 I=2,7
11790      IF(ETI(I).EQ.BLANC) GOTO 7104
11800      BETQ(I)=ETI(I)-ETI(I)/2**27*2**27
11810 7103 ETI(I)=ETI(I)-BETQ(I)
11820       IETI=6
11830      GOTO 7106
11840 7104 IETI=I-1
11850 7106 GOTO (7111,7112,7113,7114,7115,7116),IETI
11860 7111 ETIQ1=ETI(2)
11870      GOTO 7118
11880 7112 ETIQ1=ETI(2)+ETI(3)/2**9
11890       GOTO 7118
11900 7113 ETIQ1=ETI(2)+ETI(3)/2**9+ETI(4)/2**18
11910       GOTO 7118
11920 7114 ETIQ1=ETI(2)+ETI(3)/2**9+ETI(4)/2**18+ETI(5)/2**27
11930 7118 ETIQ2=BLANC4
```

```
      GOTO 7119
 7115 ETIQ2=ETI(6)
      GOTO 7121
 7116 ETIQ2=ETI(6)+ETI(7)/2**9
 7121 ETIQ1=ETI(2)+ETI(3)/2**9+ETI(4)/2**18+ETI(5)/2**27
 7119 CONTINUE
      CALL CRSYMB (ETIQ1,ETIQ2,NPROG1,NPROG2,NOMMAC,CPTORD,
     &NBINST,PLASYM,NBPARA,ERR,TABSYM,100,50)
      GOTO 2222
 5000 CONTINUE
      DEMAND=ETI(2)+ETI(3)/2**9+ETI(4)/2**18+ETI(5)/2**27
      CALL COMMAND (ORDRE ,DEMAND,OSYM,OBIA,OBIR,OTSYM)
 2222 CONTINUE
      IF(ERR.EQ.1) GOTO 3222
      CALL SOPBIN (VAR,VAR2,PREVAR,PREVAD,ADRUEX,CO,SUN,ZERO,
     &JRTRE)
 3222 CONTINUE
      IF(FIN.EQ.0) GOTO 3333
      CALL NIF03(BIN)
      STOP
      END
C
C
C
C
C
C*************************
C ETUDE DES SOUS PROGAMMES
C*************************
C ETUDE INSTRUCTION * REFM NOM;
C***************************
      SUBROUTINE REFMAC(SYMB1,SYMB2,NOMMAC,TMACRP,IMACRP,PMACRP,
     &FMACRP,TMACRO,IMACRO,PMACRO,FMACRO,BINREL,NBINST,NINSTM,NIVMAC,
     &CPTORD,CPTORM,N,M,L,LP,LO)
      INTEGER SYMB1,SYMB2,NOMMAC(2),PMACRP,PMACRO,CPTORD,CPTORM
      INTEGER TMACRP(100,4),TMACRO(100,4),FMACRP(LP,32),FMACRO(LO,32)
     &,BINREL(L,32)
      INTEGER BLANC
      DATA BLANC /1H /
      IF(NOMMAC(1).EQ.BLANC) GO TO 10
      I=1
   25 IF(SYMB1.EQ.TMACRP(I,1)) GO TO 20
   35 IF(I.EQ.IMACRP) GO TO 15
      I=I+1
      GO TO 25
   20 IF(SYMB2.EQ.TMACRP(I,2)) GO TO 30
      GO TO 35
   15 I=1
   50 IF(SYMB1.EQ.TMACRO(I,1)) GO TO 40
   60 IF(I.EQ.IMACRO) GO TO 45
      I=I+1
      GO TO 50
```

```
12440   40 IF(SYMB2.EQ.TMACRO(I,2)) GO TO 55
12450      GO TO 60
12460   45 WRITE(6,11)
12470   11 FORMAT(47H*** ERREUR : LA MACRO APPELEE N'EXISTE PAS *** )
12480      GO TO 100
12490   30 PMACRP=I
12500      J=1
12510   75 DO 65 K=1,32
12520   65 BINPEL(NBINST+J,K)=PMACRP(TMACRP(PMACRP,3)+J,K)
12530      IF(J.EQ.TMACRP(PMACRP,4)) GO TO 70
12540      J=J+1
12550      GO TO 75
12560   70 CPTORD=CPTORD+TMACRP(PMACRP,4)
12570      NBINST=NBINST+TMACRP(PMACRP,4)
12580      GO TO 100
12590   55 PMACRO=I
12600      J=1
12610   90 DO 80 K=1,32
12620   80 BINPEL(NBINST+J,K)=PMACRO(TMACRO(PMACRO,3)+J,K)
12630      IF(J.EQ.TMACRO(PMACRO,4)) GO TO 85
12640      J=J+1
12650      GO TO 90
12660   85 CPTORD=CPTORD+TMACRO(PMACRO,4)
12670      NBINST=NBINST+TMACRO(PMACRO,4)
12680      GO TO 100
12690   10 I=1
12700  115 IF(SYMB1.EQ.TMACRP(I,1)) GO TO 105
12710  125 IF(I.EQ.TMACRP) GO TO 110
12720      I=I+1
12730      GO TO 115
12740  105 IF(SYMB2.EQ.TMACRP(I,2)) GO TO 120
12750      GO TO 125
12760  120 PMACRP=I
12770      NIVMAC=NIVMAC+1
12780      J=1
12790  140 DO 130 K=1,32
12800  130  PMACRP(NINSTM+J,K)=PMACRP((TMACRP(PMACRP,3)+J),K)
12810      IF(J.EQ.TMACRP(PMACRP,4)) GO TO 135
12820      J=J+1
12830      GO TO 140
12840  135 CPTORM=CPTORM+TMACRP(PMACRP,4)
12850      NINSTM=NINSTM+TMACRP(PMACRP,4)
12860      NIVMAC=NIVMAC-1
12870      GO TO 100
12880  110 I=1
12890  155 IF(SYMB1.EQ.TMACRO(I,1)) GO TO 145
12900  165 IF(I.EQ.TMACRO) GO TO 150
12910      I=I+1
12920      GO TO 155
12930  150 WRITE(6,12)
12940   12 FORMAT(46H*** ERREUR : LA MACRO APPELEE N'EXISTE PAS ***   )
12950      GO TO 100
```

```
960   145 IF(SYMB2.EQ.TMACRO(I,2)) GO TO 160
970       GO TO 165
980   160 PMACRO=I
990       NIVMAC=NIVMAC+1
1000      DO 170 J=1,TMACRO(IMACRO,4)
1010      DO 170 K=1,32
1020  170 FMACRP(NINSTM+J,K)=FMACRO(TMACRO(PMACRO,3)+J,K)
1030      CPTORM=CPTORM+TMACRO(PMACRO,4)
1040      NINSTM=NINSTM+TMACRO(PMACRO,4)
1050      NIVMAC=NIVMAC-1
1060  100 RETURN
1070      END
1080C*****************
1090C ETUDE INSTRUCTIONDEF DE MACRO
1100C****************
1110      SUBROUTINE CRMACR(SYMB1,SYMB2,TABSYM,NBPARA,TMACRP,IMACRP,
1120     &NINSTM,CPTORM,NIVMAC,NBMACRP,NOMMAC,N,M)
1130      INTEGER TABSYM(N,6),TMACRP(M,4),NOMMAC(2)
1140      INTEGER SYMB1,SYMB2,CPTORM
1150      I=1
1160    5 IF(TABSYM(I,4).EQ.SYMB1) GO TO 10
1170   25 IF(I.EQ.NBPARA) GO TO 20
1180      I=I+1
1190      GO TO 5
1200   10 IF(TABSYM(I,5).EQ.SYMB2) GO TO 15
1210      GO TO 25
1220   15 IF(TABSYM(I,1).EQ.2) GO TO 30
1230      WRITE(6,11)
1240   11 FORMAT(52H*** ERREUR : SYMB DEJA UTILISE,NE PEUT ETRE NOM MAC    )
1250      GO TO 100
1260   30 WRITE(6,12)
1270   12 FORMAT(52H*** ERREUR : SYMB DEJA DEFINI NOM MACRO GENERALE ***)
1280      GO TO 100
1290   20 I=1
1300   50 IF(SYMB1.EQ.TMACRP(I,1)) GO TO 35
1310   45 IF(I.EQ.IMACRP) GO TO 40
1320      I=I+1
1330      GO TO 50
1340   35 IF(SYMB2.NE.TMACRP(I,2)) GO TO 45
1350      WRITE(6,13)
1360   13 FORMAT(53H*** SYMB DEJA UTILISE COMME NOM DE MACRO PARTICULIERE
1370     &16HDANS CE PROG *** )
1380      GO TO 100
1390   40 TMACRP(IMACRP,1)=SYMB1
1400      TMACRP(IMACRP,2)=SYMB2
1410      IF(IMACRP.EQ.1) GO TO 55
1420      TMACRP(IMACRP,3)=NINSTM+1
1430      GO TO 60
1440   55 TMACRP(IMACRP,3)=1
1450      NOMMAC(1)=SYMB1
1460      NOMMAC(2)=SYMB2
1470   60 CPTORM=0
```

```
13480        NIVMAC=NIVMAC+1
13490        NBMACRP=NBMACRP+1
13500   100  RETURN
13510        END
13520        SUBROUTINE CRPEND (TMACRP,IMACRP,CPTORM,NOMMAC,NIVMAC,M)
13530        INTEGER TMACPP(M,4),CPTORM
13540        INTEGER BLANC
13550        DATA BLANC /1H /
13560        INTEGER NOMMAC(2)
13570        TMACRP(IMACPP,4)=CPTORM
13580        NOMMAC(1)=BLANC
13590        NOMMAC(2)=BLANC
13600        NIVMAC=NIVMAC-1
13610        IMACRP=IMACRP+1
13620        RETURN
13630        END
13640        SUBROUTINE CRPROG (SYMB1,SYMB2,TABSYM,NBPARA,TAIPRO,IPRO,
13650       &NBINST,NINSTM,NIVMAC ,N,M,ERR)
13660        INTEGER SYMB1,SYMB2
13670        INTEGER TAIPRO (N , 4) ,TABSYM(M,6 )
13680        INTEGER  IPRO,NBINST ,NINSTM ,NIVMAC
13690        INTEGER ERR
13700        INTEGER TAT
13710        CALL REMPLI(NBPARA,IPRO,TABSYM,TAIPRO,NBINST,
13720       &NINSTM,NIVMAC,N,M)
13730        WRITE (6,6001) ((TABSYM(I,J),J=1,6),I=1,4)
13740 6001   FORMAT ('TABSYM',2X,I2,4(1H*,A4),1H*,I4)
13750        WRITE (6,6008)
13760        READ (5,506) INO
13770        WRITE (6,6007) ((TABSYM(I,J),J=1,6),I=1,INO)
13780 6007   FORMAT (6(2Y,O12))
13790   506  FORMAT (I2)
13800 6008   FORMAT('INO=')
13810        I=1
13820        WRITE (6,6004) SYMB1,SYMB2
13830 6004   FORMAT ('SYMB1=',2X,A4,2X,'SYMB2=',2X,A4,2X)
13840    30  IF (SYMB1.EQ.TABSYM(I,4)) GOTO 5
13850    15  IF (NBPARA.EQ.I) GOTO 20
13860        I=I+1
13870        GOTO 30
13880     5  IF (SYMB2.EQ.TABSYM(I,5)) GOTO 10
13890        GOTO 15
13900 10     CONTINUE
13910        WRITE (6,6005) I
13920 6005   FORMAT ('W6005',I2)
13930        IF(TABSYM(I,1).EQ.1) GOTO 1010
13940        IF (TABSYM(I,1).EQ.2) GOTO 1020
13950        IF (TABSYM(I,1).EQ.4) GOTO 1030
13960        TAT=TABSYM(I,1)
13970        WRITE (6,6013) TAT
13980 6013   FORMAT ('W6013 TAT ',2X,O12)
13990        IF((TAT.EQ.14).OR.(TAT.EQ.15).OR.(TAT.EQ.16).
```

```
67T 01  08-20-75   11.697      CONVERSION DE SUPPORT        PAR BPRINT

000       .OR.(TAT.EQ.17)) GOTO 1080
010        IF((TAT.EQ.51).OR.(TAT.EQ.52).OR.(TAT.EQ.53)) GOTO 50
020        IF (TABSYM(I,1).EQ.50) GOTO 1040
030        IF((TAT.EQ.60).OR.(TAT.EQ.61).OR.(TAT.EQ.62)) GOTO 1050
040        IF (TABSYM(I,1).EQ.7) GOTO 1060
050        WRITE(6,1001)
060 1001 FORMAT ( 30H*** ERREUR :GENRE INCONNU *** )
070        GOTO  100
080   20 NBPARA =NBPARA+1
090   50 TABSYM(I,1)=1
100        TABSYM(I,2)=SYMB1
110        TABSYM(I,3)=SYMB2
120        TABSYM(I,4)=SYMB1
130        TABSYM(I,5)=SYMB2
140        WRITE (6,6011) I,(TABSYM(I,NI),NI=1,6)
150 6011 FORMAT('H6011',/,I2/,6(2X,O12))
160        J=1
170   80   IF (TAIPRO (J,1).EQ.SYMB1) GOTO  60
180   90 IF (J.EQ. IPRO ) GOTO 70
190        J=J+1
200        GOTO 80
210   60 IF (TAIPRO(J,2).EQ.SYMB2) GOTO 1070
220        GOTO 90
230 70     CONTINUE
240        WRITE (6,6014) SYMB1,SYMB2
250 6014 FORMAT('H6014',2X,'SYMB1',2X,O12,2X,'SYMB2',2X,O12)
260        TAIPRO(J,1)=SYMB1
270        TAIPRO (J,2)=SYMB2
280        TAIPRO (J,4)=NBINST
290        WRITE (6,6012) J,(TAIPRO(J,NU),NU=1,4)
300 6012 FORMAT('H6012',/,I2,/,4(2X,O12))
310        IPRO =IPRO +1
320        NINSTR =0
330        NIVMAC =0
340        GOTO 100
350 1010 WRITE (6,1011)
360 1011 FORMAT  ( 48H*** ERREUR :NOM DEJA DEFINI COMME NOM DE PROG***)
370        GOTO 100
380 1020 WRITE (6,1021)
390 1021 FORMAT (51H*** ERREUR :NOM DEJA DEFINI COMME NOM  DE MACRO***  )
400        GOTO 100
410 1030 WRITE (6,1031)
420 1031 FORMAT (53H*** ERREUR :NOM DEJA DEFINI COMME NOM DE SYMB ETIQ***)
430        GOTO 100
440 1040 WRITE (6,1041)
450 1041 FORMAT (54H*** ERREUR :NOM DE SYMB DEJA REF ET NON ENCORE UTILISE)
460        GOTO 100
470 1050 WRITE (6,1051)
480 1051 FORMAT (40H*** ERREUR :NOM DE SYMB DEJA DEFINI ***   )
490        GOTO 100
500 1060 WRITE (6,1061)
510 1061 FORMAT (54H*** ERREUR :SYMB AUQUEL UNE VAL A DEJA ETE  AFFECTEE *)
```

```
14520        GOTO 100
14530 1070 WRITE (6,1071)
14540 1071 FORMAT (53H*** ERREUR : SYMB DEJA UTILISE COMME NOM DE  PROG ***
14550        GOTO 100
14560 1080 WRITE (6,1081)
14570 1081 FORMAT (54H*** ERREUR : SYMB DEJA DEF COMME UN SYMBOLE INTERNE**
14580  100 CONTINUE
14590        WRITE (6,6009) ((TABSYM(I,J),J=1,6),I=1,INO)
14600 6009 FORMAT('TABSYM',6(2X,O12))
14610        WRITE (6,6010)((TAIPRO(I,J),J=1,4),I=1,INO)
14620 6010 FORMAT ('TAIPRO',4(2X,O12))
14630        WRITE (6,6002) ((TABSYM(I,J),J=1,6),I=1,5)
14640 6002 FORMAT ('TABSYM',2X,I2,4(A4,1H*),I4)
14650        WRITE (6,6003) ((TAIPRO(I,J),J=1,4),I=1,5)
14660 6003 FORMAT ('TAIPRO',2X,A4,2X,A4,2X,I4,2X,I4)
14670        RETURN
14680        END
14690        SUBROUTINE FINPROG (SYMB1,SYMB2,ETIQ1,ETIQ2,TAIPRO,CPTORD,NBINST
14700       &ERR,IPRO)
14710        INTEGER ETIQ1,ETIQ2,SYMB1,SYMB2
14720        INTEGER BLANC
14730        DATA BLANC /1H /
14740        INTEGER TAIPRO (N ,4)
14750        INTEGER IPRO,CPTORD,NBINST
14760        INTEGER ERR
14770        IF(ETIQ1.EQ.BLANC) GOTO 10
14780        CALL  CRSYMB (SYMB1)
14790        CPTORD =CPTORD+1
14800        NBINST =NBINST +1
14810        CALL  BRINRE(SYMB1)
14820   10 I =1
14830   30 IF (TAIPRO (I,1).EQ.SYMB1) GOTO 20
14840   50 IF (I.EQ.IPRO) GOTO  1010
14850        I=I+1
14860        GOTO 30
14870   20 IF (TAIPRO (I,2).EQ.SYMB2) GOTO 40
14880        GOTO 50
14890   40 TAIPRO (I,3) =CPTORD
14900 1010 WRITE (6,1011)
14910 1011 FORMAT (  52H***ERREUR: LE  PROG NE SE TROUVE PAS DANS TAIPRO***
14920        GOTO  100
14930  100 RETURN
14940        END
14950        SUBROUTINE  PDEF (SYMB1,SYMB2,NPROG1,NPROG2,NBPARA,TABSYM,N,M,ER
14960        INTEGER  SYMB1 ,SYMB2 ,NPROG1,NPROG2
14970        INTEGER  NBPARA ,ERR
14980        INTEGER  TABSYM(N ,6)
14990        INTEGER TAT
15000        WRITE (6,6002) SYMB1,SYMB2,NPROG1,NPROG2
15010 6002 FORMAT ('SYMB1',2X,O12,2X,'SYMB2',2X,O12,2X,
15020       &'NPROG1',2X,O12,2X,'NPROG2',2X,O12)
15030        INTEGER TAB
```

```
1040      WRITE (6,6001) ((TABSYM(I,J),J=1,6),I=1,4)
1050 6001 FORMAT('TABSYM'/,6(2X,O12))
1060      I=1
1070   30 IF (TABSYM(I,4).EQ.SYMB1) GOTO 10
1080   60 IF (NBPARA.EQ.I) GOTO 20
1090      I=I+1
1100      GOTO 30
1110   10 IF (TABSYM(I,5).EQ.SYMB2) GOTO 40
1120      GOTO 60
1130   40 IF (TABSYM(I,1).LE.4) GOTO 1010
1140      TAT=TABSYM(I,1)
1150      IF((TAT.EQ.60).OR.(TAT.EQ.61).OR.(TAT.EQ.62).OR.
1160     &(TAT.EQ.63)) GOTO 70
1170      WRITE (6,1001)
1180 1001 FORMAT(   33H*** ERREUR: SYMB  DEJA DEFINI***  )
1190      GOTO  100
1200  70  IF((TAT.EQ.51).OR.(TAT.EQ.53).OR.(TAT.EQ.52)) GOTO 80
1210      IF (TABSYM(I,1).EQ.50) GOTO  1020
1220      IF(TABSYM (I,1).EQ.77 GOTO 1030
1230      WRITE (6,1041)
1240 1041 FORMAT (31H*** ERREUR : GENRE INCONNU  *** )
1250      GOTO 100
1260 1030 WRITE (6,1031)
1270 1031 FORMAT (51H*** ERREUR :SYMB  AUQUEL ON A AFFECTE UNE VALEUR***)
1280      GOTO  100
1290 1020 WRITE (6,1021)
1300 1021 FORMAT (54H*** ERREUR : SYMB DEJA REFERENCE ,NON ENCORE UTILISE *)
1310      GOTO 100
1320 1010 WRITE (6,1011)
1330 1011 FORMAT (53H** ERREUR: SYMB DEJA DEF COMME NOM DE PROG-MACRO-SYMB
1340     &7H-ETIQ *)
1350      GOTO 100
1360   20 TABSYM(I,4)=SYMB1
1370      TABSYM(I,5)=SYMB2
1380      NBPARA =NBPARA +1
1390   80 TABSYM(I,1)= 60
1400      TABSYM(I,2)=NPROG1
1410      TABSYM(I,3)=NPROG2
1420        WRITE (6,6001)((TABSYM(I,J),J=1,6),I=1,7)
1430      WRITE (6,6003) NBPARA
1440 6003 FORMAT ('NBPARA=' ,I3)
1450  100 RETURN
1460      END
1470      SUBROUTINE   PREF (SYMB1,SYMB2,NPROG1,NPROG2,NBPARA,TABSYM,M,ERR)
1480      INTEGER ERR
1490      INTEGER  SYMB1,SYMB2,NPROG1,NPROG2
1500      INTEGER TABSYM (M ,6 )
1510      INTEGER NBPARA
1520      I= 1
1530      WRITE (6,6001) ((TABSYM(NI,NJ),NJ=1,6),NI=1,8)
1540 6001 FORMAT('TABSYM',/,6(2X,O12))
1550   30 IF ( TABSYM(I,4).EQ.SYMB1) GOTO 10
```

```
15560    50 IF ( I.EQ. NBPARA)  GOTO 20
15570       I=I+1
15580       GOTO  50
15590    10 IF (SYMB2.EQ.TABSYM(I,5))GOTO 40
15600       GOTO 50
15610   40  TAT=TABSYM(I,1)
15620       IF((TAT.EQ.1).OR.(TAT.EQ.4).OR.(TAT.EQ.51).OR.
15630      &(TAT.EQ.52).OR.(TAT.EQ.53).OR.(TAT.EQ.61).OR.
15640      &(TAT.EQ.62).OR.(TAT.EQ.63)) GOTO 100
15650       IF (TABSYM(I,1).EQ.2)  GOTO 1010
15660       IF (TABSYM(I,1).EQ. 50)  GOTO 1020
15670       IF (TABSYM(I,1).EQ. 60) GOTO 1030
15680       IF((TAT.EQ.14).OR.(TAT.EQ.15).OR.(TAT.EQ.16).
15690      &OR.(TAT.EQ.17)) GOTO 1040
15700       IF  (TABSYM (I,1).EQ.7)  GOTO  1050
15710       WRITE (6,1001)
15720 1001  FORMAT ( 34H*** ERREUR  GENRE NON  DEFINI ***    )
15730       GOTO  100
15740 1010  WRITE (6,1011)
15750 1011  FORMAT ( 46H*** ERREUR :SYMB DEJA DEF  COMME NON MACRO ***  )
15760       GOTO  100
15770 1020  WRITE  (6,1021)
15780 1021  FORMAT ( 41H***ERREUR   : SYMB DEJA  REFERENCE ET NON
15790      &20HENCORE UTILISE  ***  )
15800       GOTO  100
15810 1031  FORMAT (45H***ERREUR   :SYMB DEJA DEF ET NON UTILISE***   )
15820 1030  WRITE (6,1031)
15830       GOTO  100
15840 1040  WRITE (6,1041)
15850 1041  FORMAT ( 41H*** ERREUR :SYMB DEJA DEF COMME INTERNE *  )
15860       GOTO  100
15870 1050  WRITE (6,1051)
15880 1051  FORMAT  (52H*** ERREUR  SYMB AUQUEL UNE VAL A DEJA ETE AFFECTEE*
15890       GOTO  100
15900    20 TABSYM (I,1) = 50
15910       TABSYM (I,2) =NPROG1
15920       TABSYM (I,3) =NPROG2
15930       TABSYM (I,4) =SYMB1
15940       TABSYM (I,5) =SYMB2
15950       NBPARA=NBPARA +1
15960       WRITE (6,6002) (TABSYM(I,NI),NI=1,6)
15970 6002  FORMAT('TABSYM',/,6(2X,O12))
15980   100 RETURN
15990       END
16000       SUBROUTINE   CREGAL (SYMB1,SYMB2,NPROG1,NPROG2,VALEUR,NBPARA,
16010      &TABSYM,M,ERR)
16020       INTEGER  NBPARA
16030       INTEGER  SYMB1,SYMB2,NPROG1,ERR,NPROG2,VALEUR
16040       INTEGER  TABSYM (M ,6 )
16050       INTEGER  ERR
16060       I=1
16070    30 IF (TABSYM(I,4).EQ.SYMB1 ) GOTO 10
```

```
57T 01  08-20-75   11.697   CONVERSION DE SUPPORT        PAR BPRINT

   50 IF (I.EQ.NBPARA )GOTO 20
      I=I+1
      GOTO 30
   10 IF (TABSYM(I,5).EQ.SYMB2 ) GOTO 40
      GOTO 50
   40 IF (TABSYM (I,1).EQ.7) GOTO 60
      IF (TABSYM (I,1).EQ.16) GOTO 70
      TAT=TABSYM(I,1)
      IF((TAT.EQ.14).OR.(TAT.EQ.15).OR.(TAT.EQ.16)) GOTO 80
      IF((TAT.EQ.1).OR.(TAT.EQ.2).OR.(TAT.EQ.4)) GOTO 1010
      IF((TAT.EQ.50).OR.(TAT.EQ.51).OR.(TAT.EQ.52).OR.
     &(TAT.EQ.53).OR.(TAT.EQ.60).OR.(TAT.EQ.61).OR.(TAT.EQ.62)
     &.OR.(TAT.EQ.63)) GOTO1040
      WRITE (6,1001)
 1001 FORMAT (28H***ERREUR :GENRE INCONNU *** )
      GOTO 100
   20 TABSYM (I,1)=7
      TABSYM (I,2)=NPROG1
      TABSYM (I,3)=NPROG2
      TABSYM (I,4)=SYMB1
      TABSYM (I,5)=SYMB2
      TABSYM (I,6)=VALEUR
      NBPARA=NBPARA+1
      GOTO 100
   60 J=1
      GOTO 90
   70 J=2
      GOTO 90
   80 J=3
   90 IF((TABSYM(I,2).EQ.NPROG1).AND.(TABSYM(I,3).EQ.NPROG2))GOTO 110
      I=NBPARA
      GOTO 20
  110 GOTO (1020,120,1030) , 7
 1020 WRITE (6,1021)
 1021 FORMAT (55H***ERREUR:SYMB DU PROG AUQUEL UNE VAL A DEJA ETE  AFFEC
     &9HTEE  *** )
      GOTO  100
 1030 WRITE (6,1031)
 1031 FORMAT (49H***ERREUR : SYMB INTERNE DEJA DEF DS LE PROG ***   )
      GOTO  100
 1010 WRITE(6,1011)
 1011 FORMAT(50H***ERREUR:SYMB DEF COMME NOM PROG,NOMMACRO
     &       )
      GOTO100
  120 GOTO 20
 1040 WRITE (6,1041)
 1041 FORMAT (40H***ERREUR :SYMB EXT DEJA REF OU DEF ***   )
  100 RETURN
      END
      SUBROUTINE CRSYMB (SYMB1,SYMB2,NPROG1,NPROG2,NOMMAC,CPTORD,NBINST
     &,PLASYM,NBPARA,ERR,TABSYM,N,M )
      INTEGER SYMB1,SYMB2,NPROG1,NPROG2,NOMMAC(2)
```

```
16600        INTEGER BLANC
16610        DATA BLANC /1H /
16620        INTEGER  CPTORD ,NINSTR ,PLASYM(N ) ,NBPARA ,ERR
16630        INTEGER  TABSYM (M  ,6 )
16640        ERR =0
16650        IF (NOMMAC(1).NE.BLANC)GOTO 500
16660        GOTO 501
16670  500 WRITE  (6,1081)
16680 1081 FORMAT (  35H***ERREUR :SYMB DEF  DS UNE  MACRO )
16690 501  CONTINUE
16700        I=1
16710   30 IF (TABSYM(I,4).EQ.SYMB1) GOTO 10
16720   50 IF (I.EQ.NBPARA) GOTO 20
16730        I=I+1
16740        GOTO  30
16750   10 IF (TABSYM (I,5).EQ. SYMB2 ) GOTO 40
16760        GOTO 50
16770  40  TAT=TABSYM(I,1)
16780        IF((TAT.EQ.15).OR.(TAT.EQ.16).OR.(TAT.EQ.17)) GOTO 110
16790        IF (TABSYM (I,1).EQ.14 ) GOTO 120
16800        IF((TAT.EQ.60).OR.(TAT.EQ.61).OR.(TAT.EQ.62).OR.
16810       &(TAT.EQ.63)) GOTO 130
16820        IF((TAT.EQ.50).OR.(TAT.EQ.51).OR.(TAT.EQ.52).OR.
16830       &(TAT.EQ.53)) GOTO 1010
16840        IF (TABSYM (I,1).EQ.7) GOTO 1020
16850        IF((TAT.EQ.1).OR.(TAT.EQ.2).OR.(TAT.EQ.4)) GOTO 1030
16860        WRITE (6,1001)
16870 1001 FORMAT (28H*** ERREUR : GENRE INCONNU *)
16880        GOTO 100
16890 1030 WRITE  (6,1031)
16900 1031 FORMAT (62H*** ERREUR : SYMB DEJA DEF  COMME NOM  PROG,MACRO OU
16910       &ETIQ   ***  )
16920        GOTO  100
16930 1020 WRITE  (6,1021)
16940 1021 FORMAT (58H*** ERREUR : SYMB  AUQUEL UNE VAL A DEJA ETE AFFECTE
16950       &** )
16960        GOTO  100
16970 1010 WRITE (6,1011)
16980 1011 FORMAT (92H*** ERREUR : SYMB EXT REF  ,PAS ENCORE  DEF ,NE PEUT
16990       &PAS ETRE EN  ZONE ETIQ  DS CE  PROG  *** )
17000        GOTO  100
17010  110 J=1
17020        GOTO  150
17030  120 J=2
17040        GOTO  150
17050  130 J=3
17060  150 IF((TABSYM(I,2).EQ.NPROG1).AND.(TABSYM(I,3).EQ.NPROG2 ))
17070       &GOTO  160
17080        GOTO  (210,220,230),J
17090  210 WRITE  (6,1041)
17100 1041 FORMAT (52H***ERREUR :NOM SYMB INT UTILISE DS UN AUTRE PROG NON
17110       &13HTERMINE  ***  )
```

```
      GOTO  100
  220 I = NBPARA
      GOTO  20
  230 WRITE (6,1051)
 1051 FORMAT (80H*** ERREUR: SYMB DEF DS UN PROG , AUCUNE VAL NE LUI
     &A ETE AFFECTEE   ***                                          )
      GOTO  100
  160 GOTO  (310,320,330),J
  310 TABSYM (I,1)=14
      TABSYM (I,6)=CPTORD
      PLASYM(NBINST)=I
      GOTO  200
  320 WRITE (6,1061)
 1061 FORMAT(52H*** ERREUR : SYMB ETIQ DEJA UTILISE DS CE PROG **     )
      GOTO 100
  330 TABSYM(I,1)=4
      TABSYM(I,6)=CPTORD
      PLASYM(NBINST)=I
      GOTO 200
  20  TABSYM(I,1)=NBPARA
      TABSYM(I,2)=NPROG1
      TABSYM(I,3)=NPROG2
      TABSYM(I,4)=SYMB1
      TABSYM(I,5)=SYMB2
      TABSYM(I,6)=CPTORD
      NBPARA=NBPARA+1
      PLASYM(NBINST)=I
      GOTO 200
  100  ERR=1
  200 RETURN
      END
      SUBROUTINE SADRUP(SYMB1,SYMB2,NPROG1,NPROG2,NINSTR,NBPARA,
     &TABSYM,PLASYM,ERR,N,M)
      INTEGER NPROG1,NPROG2,SYMB1,SYMB2,ERR,NINSTR
      INTEGER TABSYM(M,6),PLASYM(N)
      I=1
  30  IF(TABSYM(I,4).EQ.SYMB1) GOTO 10
  50  IF(I.EQ.NBPARA) GOTO 20
      I=I+1
      GOTO 30
  10  IF(TABSYM(I,5).EQ.SYMB2) GOTO 40
      GOTO 50
  40  IF((TABSYM(I,2).EQ.NPROG1).AND.(TABSYM(I,3).EQ.NPROG2))
     &GOTO 60
      TAT=TABSYM(I,1)
      IF((TAT.EQ.14).OR.(TAT.EQ.15).OR.(TAT.EQ.16).OR.
     &(TAT.EQ.17)) GOTO 70
  210 TAT=TABSYM(I,1)
      IF((TAT.EQ.60).OR.(TAT.EQ.61).OR.(TAT.EQ.63)) GOTO 80
      IF(TAT.NE.62) GOTO 90
      TABSYM(I,1)=63
      GOTO 80
```

```
17640  90  TAT=TABSYM(I,1)
17650      IF((TAT.EQ.50).OR.(TAT.EQ.51).OR.(TAT.EQ.53)) GOTO 80
17660      IF(TAT.NE.52) GOTO 110
17670      TABSYM(I,1)=55
17680      GOTO 80
17690 110  IF(TABSYM(I,1).NE.4) GOTO 120
17700      PLASYM(NINSTR)=I
17710      GOTO 200
17720 120  IF(TABSYM(I,1).NE.7) GOTO 130
17730      WRITE(6,1001)
17740 1001 FORMAT(50H***ERREUR:SYMB AUQUEL UNE VAL A ETE AFFECTEE*        )
17750      GOTO  100
17760  130 IF  (TABSYM (I,1).NE.1 ) GOTO  140
17770      PLASYM (NINSTR) = I
17780      GOTO 200
17790  140 IF  (TABSYM(I,1).NE.2)  GOTO 150
17800      WRITE (6,1011)
17810 1011 FORMAT ( 43H***ERREUR :SYMBDEJA DEF COMME NOM MACRO *** )
17820      GOTO  100
17830  150 WRITE  (6,1021)
17840 1021 FORMAT ( 31H*** ERREUR : GENRE  INCONNU *** )
17850      GOTO  100
17860   60 TAT=TABSYM(I,1)
17870      IF((TAT.EQ.14).OR.(TAT.EQ.15).OR.(TAT.EQ.17)) GOTO 80
17880      IF  (TABSYM(I,1).EQ.16) GOTO   210
17890      GOTO 80
17900   70 I=NBPARA
17910   20 TABSYM (I,1) =15
17920      TABSYM (I,2)=NPROG1
17930      TABSYM (I,3)=NPROG2
17940      TABSYM (I,4)= SYMB1
17950      TABSYM (I,5)= SYMB2
17960      NBPARA = NBPARA +1
17970   80 PLASYM(NINSTR) = I
17980      GOTO  200
17990  100 ERR =1
18000  200 RETURN
18010      END
18020      SUBROUTINE  SCHVAL (SYMB1,SYMB2,NPROG1,NPROG2,NINSTR,NBPARA,
18030     &TABSYM.PLASYM.N,M,ERR)
18040      INTEGER  TABSYM (M ,6) ,PLASYM (N ) ,
18050     &NBPARA.NINSTR,NPROG1,NPROG2,SYMB1,SYMB2
18060 6003 FORMAT('INB=NB DE LIGNE DE TABSYM DEMANDEES')
18070      WRITE (6,6003)
18080      READ (5,5001) INB
18090 5001 FORMAT(I2)
18100      INTEGER  ERR
18110      INTEGER TAT
18120      WRITE (6,6002) ((TABSYM(I,J),J=1,6),I=1,INB)
18130 6002 FORMAT('W6002',/,6(2X,O12)
18140     &   )
18150      I=1
```

```
      WRITE (6,6001)  SYMB1,SYMB2
6001   FORMAT ('W6001',2(2X,O12))
   30 IF (TABSYM (I,4).EQ.SYMB1) GOTO 10
   40 IF (I.EQ.NBPARA ) GOTO 20
      I=I+1
      GOTO 30
   10 IF (TABSYM(I,5).NE.SYMB2) GOTO 40
      IF ((TABSYM(I,4).EQ.NPROG1).AND.(TABSYM(I,5).EQ.NPROG2)) GOTO 80
      TAT=TABSYM(I,1)
      IF((TAT.NE.14).OR.(TAT.NE.15).OR.(TAT.NE.16).OR.
     &(TAT.NE.17)) GOTO 70
   60 I =NBPARA
      GOTO 20
   80 TAT=TABSYM(I,1)
      IF((TAT.EQ.14).OR.(TAT.EQ.16).OR.(TAT.EQ.17)) GOTO 110
   90 IF (TABSYM (I,1).EQ.15) GOTO 120
  70  TAT=TABSYM(I,1)
      IF((TAT.EQ.60).OR.(TAT.EQ.62).OR.(TAT.EQ.63)) GOTO 110
      IF (TABSYM(I,1).NE.61) GOTO 130
      TABSYM(I,1) =63
      GOTO 110
  120 TABSYM (I,1) = 17
      GOTO 110
130   TAT=TABSYM(I,1)
      IF((TAT.EQ.50).OR.(TAT.EQ.52).OR.(TAT.EQ.53)) GOTO 60
      IF  (TABSYM (I,1).NE.51 ) GOTO  140
      TABSYM (I,1) =53
      GOTO  60
  140 IF (TABSYM (I,1).EQ.4)  GOTO 110
      IF (TABSYM (I,1).EQ.7) GOTO  150
      IF (TABSYM (I,1).EQ.1) GOTO  110
      IF (TABSYM (I,1).EQ.2) GOTO  160
      WRITE (6,1001)
1001  FORMAT( 29H***ERREUR : GENRE INCONNU *** )
      GOTO 100
  160 WRITE (6,1011)
1011  FORMAT ( 33H*** ERREUR :SYMB NOM DE MACRO *** )
      GOTO 100
   20 TABSYM (I,1)=16
      TABSYM (I,2)=  NPROG1
      TABSYM (I,2)=NPROG2
      TABSYM (I,3)=SYMB1
      TABSYM (I,4)=SYMB2
      NBPARA= NBPARA +1
  110 PLASYM (NINSTR)= I
      GOTO  200
  150 I=NBPARA
      GOTO 20
100   ERR=1
200   CONTINUE
      WRITE (6,6002) ((TABSYM(IO,IL),IL=1,6),IO=1,INB)
      RETURN
```

```
18680        END
18690        SUBROUTINE REMPLI(NBPARA,IPRO,TABSYM,TAIPRO,NBINST,
18700       &NINSTM,NIVMAC,N,M)
18710        INTEGER IPRO,TAIPRO(N,4),TABSYM(M,6)
18720        READ (5,501) MA
18730   501  FORMAT(I1)
18740        IF(MA.EQ.1) GOTO 100
18750        DO 10 I=1,M
18760        DO 10 J=1,6
18770 10     TABSYM(I,J)=0
18780        DO 20 I=1,N
18790          DO 20 J=1,4
18800 20     TAIPRO(I,J)=0
18810        WRITE(6,601)
18820        READ (5,1) NBPARA
18830        WRITE(6,602)
18840 602    FORMAT('NBINST')
18850        READ (5,1) NBINST
18860        WRITE (6,603)
18870 603    FORMAT ('NINSTM')
18880        READ (5,1) NINSTM
18890        WRITE(6,604)
18900        READ (5,1) NIVMAC
18910        WRITE(6,605)
18920 605    FORMAT ('IPRO')
18930        READ (5,1) IPRO
18940   1    FORMAT (I2)
18950        WRITE (6,606) NBPARA,NBINST,NINSTM,NIVMAC,IPRO
18960 606    FORMAT(5(2X,I2))
18970        WRITE (6,607)
18980 607 FORMAT('TABSYM',I2,4(1X,A4),1X,I4' )
18990        WRITE (6,16)
19000   16   FORMAT ('IA=')
19010        READ (5,17) IA
19020   17   FORMAT (I2)
19030        II=1
19040 30     CONTINUE
19050        READ (5,2) (TABSYM(IA,I),I=1,6)
19060        IF(II.EQ.IA) GOTO 40
19070        II=II+1
19080        GOTO 30
19090   40   CONTINUE
19100        WRITE (6,608)
19110 608    FORMAT('TAIPRO,2(A4,1X),2(I4,1X)')
19120        READ(5,3) (TAIPRO(1,I),I=1,4)
19130 2      FORMAT(2X,I2,1H*,4(A4,1H*),I4,1H*)
19140   3    FORMAT (2X,2(A4,1H*),2(I4,1H*))
19150        WRITE (6,609)
19160 609    FORMAT ('TABSYM')
19170        WRITE (6,2) ((TABSYM(I,J),J=1,6),I=1,2)
19180        WRITE (6,3) ((TAIPRO(I,J),J=1,4),I=1,2)
19190 601    FORMAT ('NBPARA')
```

```
267T 01  08-20-75   11.697      CONVERSION DL SUPPORT        PAR BPRINT

9200 100  CONTINUE
9210 604  FORMAT('NIVMAC')
9220      RETURN
9230      END
9240      SUBROUTINE COMMAND (ORDRE,DEMAND,OSYM,OBIA,OBIR,OTSYM)
9250      INTEGER ORDRE (4),DEMAND
9260      INTEGER OSYM,OBIA,OBIR,OTSYM
9270       OSYM=0
9280      OBIA=0
9290      OBIR=0
9300      OTSYM=0
9310      DO 10 I=1,4
9320      IF(ORDRE(I).EQ.DEMAND) GOTO 20
9330 10   CONTINUE
9340      IF(DEMAND.NE.ORDRE(4)) GOTO 1001
9350 20   CONTINUE
9360      GOTO (101,102,103,104) ,I
9370 101  OSYM=1
9380      GOTO 110
9390 102   OBIA=1
9400      GOTO110
9410 103   OBIR=1
9420       GOTO 110
9430 104  OTSYM=1
9440 110  CONTINUE
9450 1001 WRITE(6,1)
9460 1    FORMAT('ERREUR COMMANDE INCONNUE:RETAPER LES CARES'
9470     &'COMMANDES************'  )
9480      RETURN
9490      END
9500      SUBROUTINE DELBIN(VALDEC,VAR1,VAR2,VBL,VZE,VUN)
9510      INTEGER VBL,VZE,VUN
9520       INTEGER VALDEC,VAR1,VAR2
9530      VAR1=0
9540      VAR2=0
9550      I=0
9560      IF(VALDEC.EQ.0) GOTO 60
9570  40  CONTINUE
9580      IN=VALDEC-(VALDEC/2)*2
9590      IF(IN.EQ.0) GOTO 10
9600      IF(I.GT.3) GOTO 20
9610     VAR1=VAR1+VUN*2**(I*9)
9620      GOTO25
9630  20  IF (I.GT.7) GOTO 30
9640      VAR2=VAR2+VUN*2**((I-4)*9)
9650      GOTO25
9660 10    IF(I.GT.3) GOTO 15
9670      VAR1=VAR1+VZE*2**(I*9)
9680      GOTO 25
9690  15  IF(I.GT.7) GOTO 30
9700      VAR2=VAR2+VZE*2**((I-4)*9)
9710  25  CONTINUE
```

```
19720        VALDEC=VALDEC/2
19730        I=I+1
19740        GOTO 40
19750   50   CONTINUE
19760       GOTO 70
19770   60   VAR1=VZE*2**27
19780        VAR2=VZE*2**27
19790   70   CONTINUE
19800        RETURN
19810        END
19820        SUBROUTINE SORBIN(VAR,VAR2,PREVAR,PREVAD,ADRUEX,CO,SUM,
19821       &ZERO,JFIN)
19830        INTEGER VAR(32),PREVAR,PREVAD,VAR2(11),CO
19840        INTEGER SUM,ZERO
19850        INTEGER SOR
19860        INTEGER S1,S2,S3,S4
19870        SOR=0
19880        INTEGER SOR1,SOR2,SOR3,SOR4
19890        INTEGER CH2(32)
19900        INTEGER SO(4)
19902        INTEGER BINREL(50,32)
19910        INTEGER UN
19920        UN=SUM
19930        INTEGER CH(32)
19940        INTEGER BIN
19950        CH(1)=PREVAR/2**27*2**27
19960        CH(2)=VAR(1)/2**27*2**27
19970        CH(3)=(VAR(1)/2**18*2**18-CH(2))*2**9
19980        CH(4)=(VAR(1)/2**9*2**9-CH(2)-CH(3)/2**9)*2**18
19990        CH(5)=(VAR(1)-CH(2)-CH(3)/2**9-CH(4)/2**18)*2**27
20000        CH(6)=VAR(2)/2**27*2**27
20010        CH(7)=(VAR(2)/2**18*2**18-CH(6))*2**9
20020        CH(8)=VAR(3)/2**27*2**27
20030        CH(9)=(VAR(3)/2**18*2**18-CH(8))*2**9
20040        CH(10)=VAR(4)/2**27*2**27
20050        CH(11)=VAR(5)/2**27*2**27
20060        CH(12)=(VAR(5)/2**18*2**18-CH(11))*2**9
20070        IF(CO.EQ.1) GOTO 110
20080        IF(CO.EQ.3) GOTO 120
20090        CH(13)=PREVAD/2**27*2**27
20100        CH(14)=VAR(6)/2**27*2**27
20110        CH (15)=(VAR(6)/2**18*2**18-CH(14))*2**9
20120        CH(16)=(VAR(6)/2**9*2**9-CH(14)-CH(15)/2**9)*2**18
20130        CH(17)=(VAR(6)-CH(14)-CH(15)/2**9-CH(16)/2**18)*2**27
20140        CH(18)=ZERO/2**27*2**27
20150        CH(19)=VAR(9)/2**27*2**27
20160        CH(20)=(VAR(9)/2**18*2**18-CH(19))*2**9
20170        CH(21)=ZERO/2**27*2**27
20180        IF(ADRUEX.EQ.1) GOTO 210
20190        GOTO 115
20200  110   CH(13)=VAR(6)/2**27*2**27
20210        CH(14)=(VAR(6)/2**18*2**18-CH(13))*2**9
```

```
0220        CH(15)=(VAR(6)/2**9*2**9-CH(13)-CH(14)/2**9)*2**18
0230        CH(16)=(VAR(6)-CH(13)-CH(14)/2**9-CH(15)/2**18)*2**27
0240        CH(17)=VAR2(6)/2**27*2**27
0250        CH(18)=(VAR2(6)/2**18*2**18-CH(17))*2**9
0260        CH(19)=(VAR2(6)/2**9*2**9-CH(17)-CH(18)/2**9)*2**18
0270        CH(20)=(VAR2(6)-CH(17)-CH(18)/2**9-CH(19)/2**18)*2**27
0280        IF(ADPUEY.EQ.1) GOTO 210
0290        CH(21)=ZERO/2**27*2**27
0300        GOTO 115
0310 120    CH(13)=VAR(6)/2**27*2**27
0320        CH(14)=(VAR(6)/2**18*2**18-CH(13))*2**9
0330         CH(15)=(VAR(6)/2**9*2**9-CH(13)-CH(14)/2**9)*2**18
0340        CH(16)=(VAR(6)-CH(13)-CH(14)/2**9-CH(15)/2**18)*2**27
0350        CH(17)=VAR(7)/2**27*2**27
0360        CH(18)=VAR(8)/2**27*2**27
0370        CH(19)=VAR(9)/2**27*2**27
0380        CH(20)=VAR(10)/2**27*2**27
0390      CH(21)=VAR2(8)/2**27*2**27
0400        GOTO 115
0410 115    CONTINUE
0420         CH(22)=VAR(11)/2**27*2**27
0430        CH(23)=(VAR(11)/2**18*2**18-CH(22))*2**9
0440        CH(24)=(VAR(11)/2**9*2**9-CH(22)-CH(23)/2**9)*2**18
0450         CH(25)=(VAR(11)-CH(22)-CH(23)/2**9-CH(24)/2**18)*2**27
0460        CH(26)=VAR(12)/2**27*2**27
0470        CH(27)=(VAR(12)/2**18*2**18-CH(26))*2**9
0480        CH(28)=VAR(13)/2**27*2**27
0490 310  CONTINUE
0500        CH(29)=VAR(14)/2**27*2**27
0510        CH(30)=VAR(15)/2**27*2**27
0520        CH(31)=VAR(16)/2**27*2**27
0530        CH(32)=(VAR(16)/2**18*2**18-CH(31))*2**9
0540        GOTO 410
0550 210    CONTINUE
0560        CH(21)=VAR(11)/2**27*2**27
0570        CH(22)=(VAR(11)/2**18*2**18-CH(21))*2**9
0580        CH(23)=(VAR(11)/2**9*2**9-CH(21)-CH(22)/2**9)*2**18
0590        CH(24)=(VAR(11)-CH(21)-CH(22)/2**9-CH(23)/2**18)*2**27
0600        CH(25)=VAR2(11)/2**27*2**27
0610        CH(26)=(VAR2(11)/2**18*2**18-CH(25))*2**9
0620        CH(27)=(VAR2(11)/2**9*2**9-CH(25)-CH(26)/2**9)*2**18
0630        CH(28)=(VAR2(11)-CH(25)-CH(26)/2**9-CH(27)/2**18)*2**27
0640        GOTO 310
0650 410    CONTINUE
0660         DO 10 I=1,32
0670        IF(CH(I).EQ.UN) GOTO 20
0680        CH2(I)=0
0690        GOTO 15
0700 20     CH2(I)=1
0710  15    CONTINUE
0720 10     CONTINUE
0730        SOR=CH2(1)*2**7+CH2(2)*2**6+CH2(3)*32+CH2(4)*16
```

```
20740      8+CH2(5)*8+CH2(6)*4+CH2(7)*2+CH2(8)
20750       SOR2=CH2(9)*2**7+CH2(10)*2**6+CH2(11)*2**5+CH2(12)*2**4
20760      8+CH2(13)*2**3+CH2(14)*4+CH2(15)*2+CH2(16)
20770       SOR3=CH2(17)*2**7+CH2(18)*64+CH2(19)*32+CH2(20)*16
20780      8+CH2(21)*8+CH2(22)*4+CH2(23)*2+CH2(24)
20790       SOR4=CH2(25)*2**7+CH2(26)*2**6+CH2(27)*32+CH2(28)*16+
20800      8CH2(29)*8+CH2(30)*4+CH2(31)*2+CH2(32)
20803       CALL STRTRE(CH2,BINREL,50,JFIN)
20810       S1=SOR*2**27
20820        S2=SOR2*2**18
20830       S3=SOR3*2**9
20840       S4=SOR4
20850       SO(1)=S1
20860       SO(2)=S2
20870       SO(3)=S3
20880       SO(4)=S4
20890       BIN=SO(1)+SO(2)+SO(3)+SO(4)
20900       WRITE(1) BIN
20910       WRITE(6,16) BIN
20920  16   FORMAT('BINAIRE DE L INSTRUCTION',/,2X,O12)
20930       RETURN
20940       END
20950        SUBROUTINE MIFO1(BIN)
20960       INTEGER BIN
20970     INTEGER BIN2,VZL,VBL,VUN
20980       BIN=1*2
20990       WRITE(1) BIN
21000       WRITE(6,2) BIN
21010  2    FORMAT('STX',2X,O12)
21020      RETURN
21030       END
21040       SUBROUTINE MIFO2(NBINST,VAR1,VAR2,BIN,VBL,VUN,VZE)
21050       INTEGER BIN,VAR1,VAR2,NBINST
21060       INTEGER VBIN2 ,VBL,VUN,VZE
21070       WRITE(6,2)
21080      READ(5,1) NBINST
21090  2    FORMAT('NBINST FORMAT I4 ')
21100  1    FORMAT(I4)
21102       VAR1=0
21103       VAR2=0
21104       GOTO100
21120       IF(NBINST.EQ.0) GOTO 60
21130  40   CONTINUE
21140       IN=NBINST-(NBINST/2)*2
21142        WRITE(6,61) IN
21143  61   FORMAT('IN=',I4)
21150       IF(IN.EQ.0) GOTO 10
21160       IF(I.GT.7) GOTO 20
21170       VAR1=VAR1+VUN*2**I
21172       WRITE(6,62) VAR1
21173  62   FORMAT('VAR1=',2X,O12)
21180       GOTO 25
```

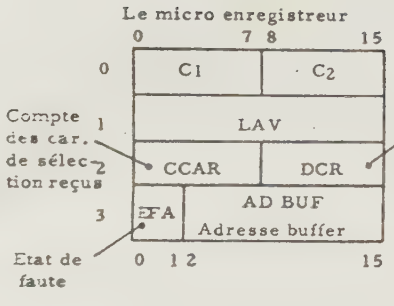
```
67T 01  08-20-75   11.697     CONVERSION DE SUPPORT        PAR PPRINT

190   20  IF(I.GT.15)  GOTO 30
200        VAR2=VAR2+VUN*2**(I-8)
202       WRITE(6,63) VAR2
203   63  FORMAT('VAR2=',2X,O12)
210       GOTO 25
220   10  IF(I.GT.7) GOTO15
230       VAR1=VAR1+VZE*2**I
232       WRITE(6,64) VAR1
233   64  FORMAT('VAR1',2X,O12)
240       GOTO25
250   15  IF(I.GT.15) GOTO 30
260       VAR2=VAR2+VZE*2**(I-8)
262       WRITE(6,65) VAR2
263   65  FORMAT('VAR2',2X,O12)
270   25  CONTINUE
280       NBINST=NBINST/2
282       IF(NBINST.EQ.0) GOTO 30
290       I=I+1
293       WRITE(6,66) I,NBINST
294   66  FORMAT('I=',I2,2X,'NBINST',2X,I3)
300       GOTO 40
310   30  CONTINUE
320       GOTO 70
330   60  VAR1=VZE
340      VAR2=VZE
350   70  CONTINUE
351  100  CONTINUE
352       IF(NBINST .GT.255) GOTO 200
353       BIN2=NBINST
354       BIN=0
355       GOTO 300
356  200 BIN=NBINST/256
357       BIN2=NBINST-BIN*256
358  300  CONTINUE
370       WRITE(1) BIN
380       WRITE(6,3) BIN
390   3   FORMAT('NBINST',2X,O12)
410       WRITE(1) BIN2
420       WRITE(6,4) BIN2
430   4   FORMAT('BIN2',2X,O12)
440       RETURN
450       END
460       SUBROUTINE MIFO3(BIN)
470       INTEGER BIN
480       BIN=128+1*2+1
490       WRITE(1) BIN
500       WRITE(6,2) BIN
510   2   FORMAT('EOT',2X,O12)
520      RETURN
530       END
540       SUBROUTINE STBIRE (CH2,BINREL,N,JPIN)
550       INTEGER CH2(32),BINREL(N,32)
```

```
21560        DO 10 I=1,32
21570        BINREL(JFIN,I)=CH2(I)
21580   10   CONTINUE
21582          WRITE(6,61)(BINREL(JFIN,I),I=1,32)
21583   61   FORMAT(32(1X,I1))
21590        JFIN=JFIN+1
21591         WRITE(6,1)
21592   1    FORMAT('NB DE LIGNE DE BINREL DESIRE=(I3)')
21593        NBLI=JFIN
21595        WRITE(2,3) ((BINREL(IO,IU),IU=1,32),IO=1,NBLI)
21596   3    FORMAT(4(8(1X,A1),1H*))
21600        RETURN
21610        END
```

ANNEXE VII

ORGANIGRAMME DES TACHES T1 ET T2

I - TACHE T1



Lecture d'un message dans la file du préprocesseur (2 mots)

V = 0 ?

ØUI → La file est vide → Retour moniteur

NØN → L'information lue est valide → Test de MP

MP = 0
- Demande d'appel
- Création d'un micro-enregisteur de n° n Sous-programme : CREMIC
- OPERANDE = adresse de voie
- Adresse de voie micro-enregistreur
- Modification du mot de commande passage dans l'état d'attente d'invitation à numéroter
- Envoi d'un message à l'UT (Demande d'appel adresse de voie)
- retour moniteur

MP = 1
- Demande de libération
- Création d'un micro-enregistreur de n° n Sous-programme : CREMIC
- OPERANDE = adresse de voie
- Adresse de voie micro-enregistreur
- Modification du mot de commande passage dans l'état d'attente de libération
- Envoi d'un message à l'UT (demande de libération adresse de voie)
- retour moniteur

MP = 2
- Demande de libération avec rappel immédiat
- (idem)

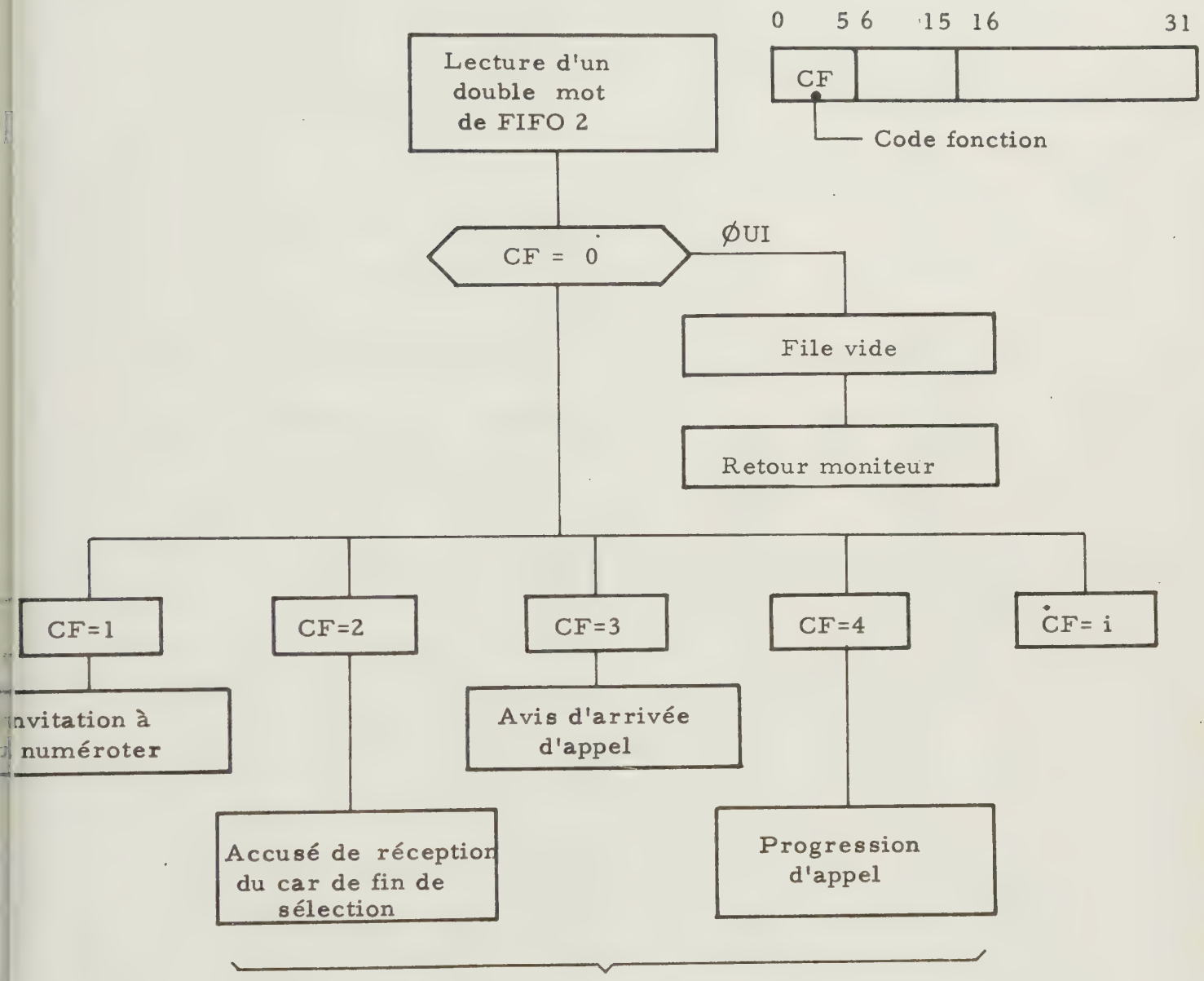
MP = 3
- Fin de sélection
- OPERANDE = PHQ numéro micro-enregistreur
- Modification du mot de commande passage dans l'état d'attente du car de progression d'appel
- Envoi d'un message à l'UT (Fin de sélection adresse de voie GCAR)

MP = 4
- Réception d'un car de sélection
- CCAR = 2 p
  - NØN → CR → CDR → CCAR CCAR+1 → Retour moniteur
- CCAR < 30
  - NØN → (CR,CDR) RDEL → Ecriture mémoire UT à l'adresse ADBUF → ADBUF ADBUF+1 → CAR ← CCAR + 1 → Retour moniteur
  - Le nombre de caractères de sélection excède la capacité d'un mini-enregistreur → Demande d'allocation d'un nouveau super-enregistreur
  - Réservation d'un super-enregistreur dans le stock de l'UMS soit SUPERi
  - Etablissement d'un lien de chaînage dans le mini-enregistreur (SUPERi) RDEL
  - Ecriture de RDEL en mémoire UT à l'adresse ADBUF
  - ADBUF SUPERi

MP = 255
- Traitement d'erreur
- Temporisation
- EFA = 3
  - NON → EFA EFA-1
  - ØUI → Envoi d'un message à l'UT → Retour moniteur

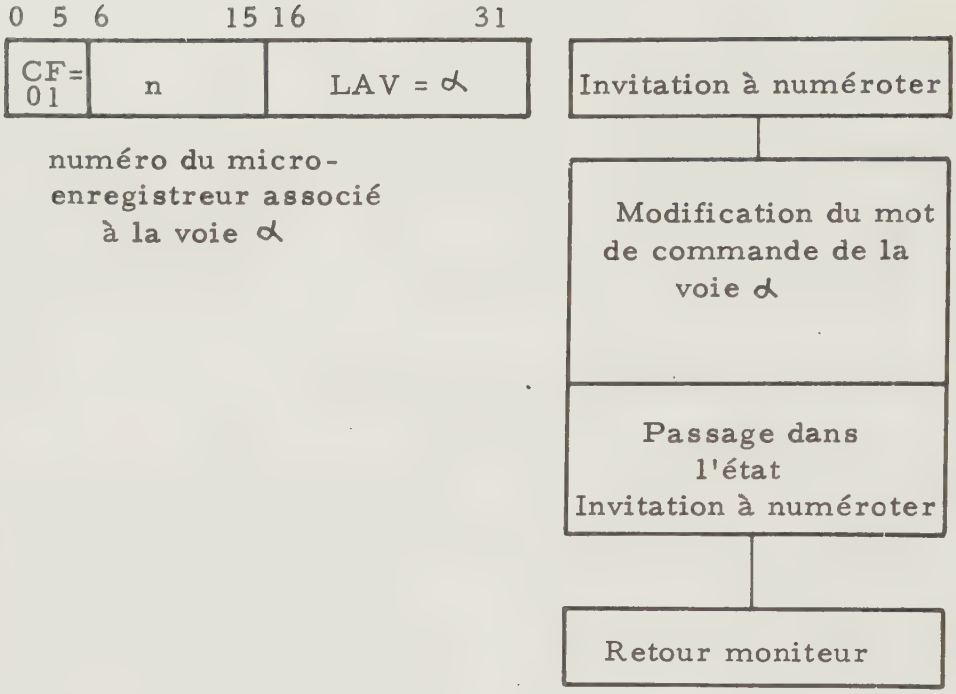
II - TACHE T2

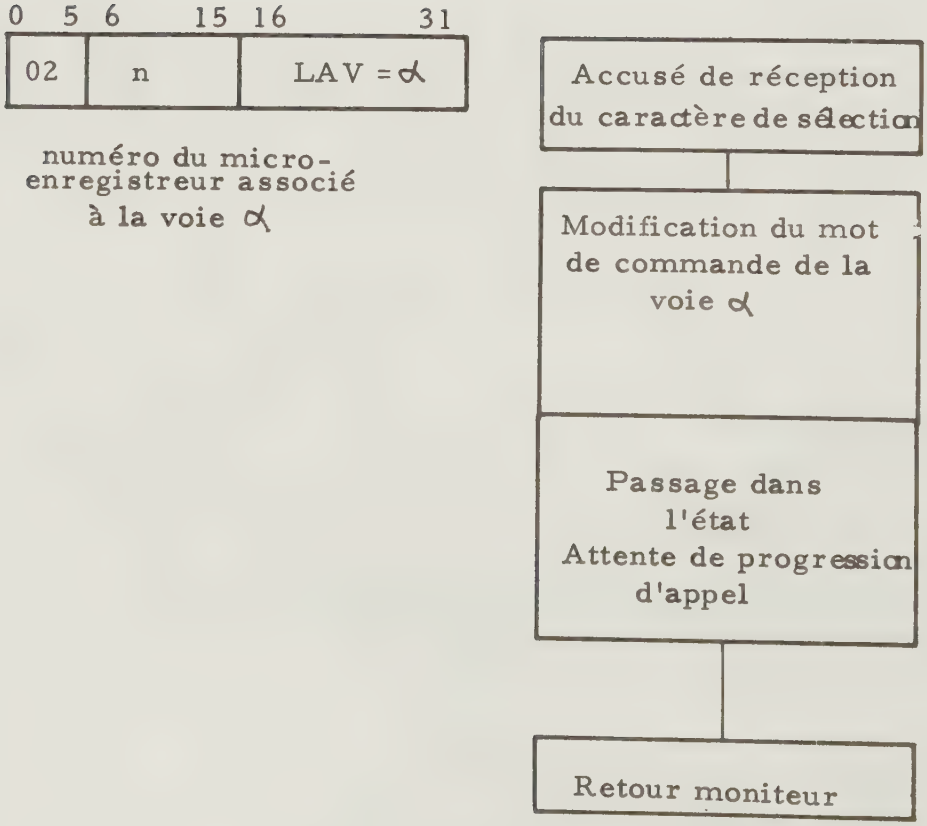
1) Organigramme général

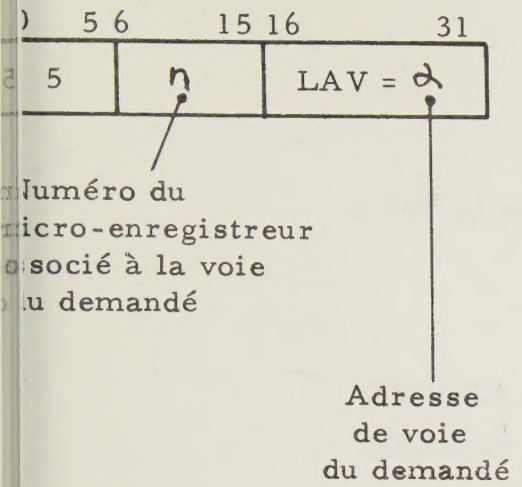


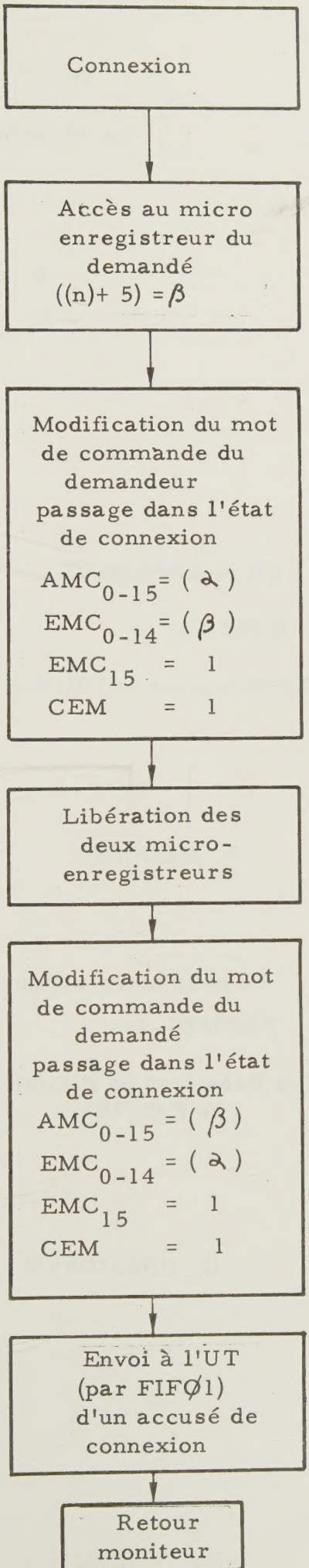
L'ensemble des commandes est décrit dans l'annexe I nous n'en donnerons ici que quelques exemples

2) Exemples de sous tâches

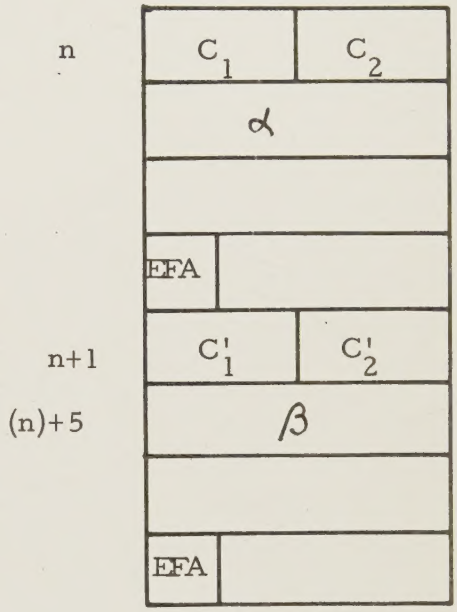








Le double micro-enregistreur



Vu :

Le Président de la Thèse,

P.Y. ARQUES

Vu :

Le Directeur de Thèse

P.Y. ARQUES

R. GERBER

VU et APPROUVE

RENNES, le

Le Directeur de l'U.E.R.

J. MEINNEL

Vu pour autorisation

RENNES, le

Le Président de l'Université
de RENNES,

C. CHAMPAUD